**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

| | |
|---|---|
| **ПОГОДЖЕНО**<br>**Декан факультету** | **ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**<br>**Завідувач кафедри інформаційних систем і технологій** |
| **_____Ігор БОЛБОТ**<br>(підпис) | **_____ Михайло ШВИДЕНКО**<br>(підпис) |
| «___» _____2026 р. | «___» _____2026 р. |

# БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

**на тему: «**Інформаційна система дистанційного керування бізнесом»
Спеціальність «122 Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні науки»

**Гарант освітньої програми**
к.т.н., доцент _____ **Роман РУДЕНСЬКИЙ**
(підпис)

**Керівник бакалаврської кваліфікаційної роботи** _____ **Роман ЗОЛОТУХА**
(підпис)
ст.викладач

**Виконав** _____ **Назар БІЛОГОЛОВИЙ**
(підпис)

**Київ-2026**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**
**Факультет інформаційних технологій**

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
інформаційних систем і технологій
(назва кафедри)

___к.е.н., доцент_____Швиденко М.З
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"6" січня 2026 р.

# З А В Д А Н Н Я

## на виконання бакалаврської кваліфікаційної роботи студенту

Білоголовому Назару Сергійовичу

Спеціальність 122 – «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи «Інформаційна система дистанційного керування бізнесом» затверджена наказом ректора НУБіП України від 06.01.2026 №46 «С»
2. Термін подання завершеної роботи на кафедру_____2026.05.22_____
(рік, місяць, число)

**3. Вихідні дані до роботи:** механізм функціонування інформаційної системи дистанційного керування бізнесом, включаючи віддалений доступ користувачів, облік операцій і ресурсів підприємства, моніторинг показників діяльності, формування звітів та аналітики через веб-інтерфейс.

**4. Перелік питань, що розглядаються:**

1. Аналіз взаємодії власника та працівників із системою дистанційного керування.
2. Проєктування інформаційного забезпечення системи управління бізнесом.
3. Розробка програмного забезпечення для контролю та аналізу діяльності.
4. Впровадження та експлуатація системи у бізнес-середовищі.

Дата видачі завдання "_06"_січня_____ 2026 р. (дата наказу)

Керівник бакалаврської кваліфікаційної роботи _____/ Золотуха Р.А./
( підпис ) (прізвище та ініціали)

Завдання прийняв до виконання:_____ / Білоголовий Н.С. /
( підпис ) (прізвище та ініціал

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

- БД: база даних.
- ІС: інформаційна система.
- ПЗ: програмне забезпечення.
- CRUD: створення, перегляд, оновлення та видалення даних.
- UI: інтерфейс користувача.
- UX: користувацький досвід.
- API: програмний інтерфейс застосунку.
- REST: архітектурний підхід до побудови вебсервісів.
- HTTP: протокол передавання гіпертексту.
- HTTPS: захищений протокол передавання гіпертексту.
- HTML: мова розмітки вебсторінок.
- CSS: каскадні таблиці стилів.
- JS: JavaScript, мова програмування для вебзастосунків.
- Node.js: середовище виконання JavaScript на сервері.
- Express.js: фреймворк Node.js для створення серверної частини вебзастосунку.
- npm: менеджер пакетів для Node.js.
- SQLite: вбудована реляційна система керування базами даних.
- SQL: мова структурованих запитів до баз даних.
- sql.js: бібліотека для роботи з SQLite у середовищі JavaScript.
- CSV: формат збереження табличних даних.
- PDF: формат електронного документа.
- URL: уніфікований покажчик ресурсу.
- bcrypt: алгоритм хешування паролів.
- session: механізм збереження стану користувача між запитами.

# ВСТУП

Динамічний розвиток малого та середнього бізнесу, поширення віддалених форматів роботи й необхідність оперативного контролю виконання управлінських рішень зумовлюють підвищені вимоги до інформаційних систем, які забезпечують дистанційне керування бізнес-процесами. У сучасних умовах керівники та менеджери підприємств потребують інструментів, що дозволяють не лише планувати роботу персоналу, а й відстежувати виконання завдань, аналізувати показники діяльності, формувати звітність та своєчасно приймати управлінські рішення незалежно від місця перебування користувачів.

Процес дистанційного керування бізнесом охоплює комплекс взаємопов’язаних дій, серед яких авторизація користувачів, розподіл ролей між власником бізнесу, менеджером і співробітником, створення та призначення завдань, контроль строків виконання, оновлення статусів, формування звітів і перегляд ключових показників діяльності. За відсутності спеціалізованого програмного забезпечення ці процеси часто виконуються за допомогою окремих таблиць, месенджерів, електронної пошти або ручного обліку, що призводить до дублювання даних, втрати актуальної інформації, ускладнення контролю та зниження ефективності управління.

Традиційні підходи до організації роботи бізнесу не завжди забезпечують цілісний інформаційний супровід управлінських процесів. Відсутність централізованої системи для обліку завдань, контролю виконавців, збереження історії виконання та аналізу результатів ускладнює оцінювання поточного стану бізнесу. Особливо актуальною ця проблема є для підприємств, у яких частина працівників виконує завдання дистанційно або працює в різних підрозділах. У таких умовах важливим стає створення єдиного інформаційного середовища, яке забезпечує прозору взаємодію між учасниками управлінського процесу.

Використання сучасних інформаційних систем, побудованих на основі клієнт-серверної архітектури, баз даних та модульного підходу до організації

програмного забезпечення, відкриває можливості для автоматизації дистанційного керування бізнесом. Такі системи дають змогу централізовано зберігати інформацію про користувачів, завдання, статуси виконання, звіти та аналітичні показники. Завдяки цьому підвищується оперативність оброблення даних, зменшується кількість помилок, спрощується контроль виконання робіт і забезпечується підтримка прийняття управлінських рішень.

**Метою кваліфікаційної роботи** є проєктування та розроблення інформаційної системи дистанційного керування бізнесом, яка забезпечує автоматизацію процесів планування, призначення, контролю виконання завдань, формування звітності та аналізу показників діяльності підприємства.

Для досягнення поставленої мети в роботі передбачається розв’язання таких **завдань:**

1. дослідити предметну область дистанційного керування бізнесом та визначити основні проблеми організації управлінських процесів;
2. проаналізувати існуючі програмні рішення для керування завданнями, персоналом і бізнес-показниками;
3. визначити функціональні, нефункціональні та безпекові вимоги до інформаційної системи дистанційного керування бізнесом;
4. спроєктувати логічну структуру системи, базу даних та взаємозв’язки між основними сутностями предметної області;
5. розробити UML-моделі, що відображають основні сценарії роботи користувачів, послідовність виконання операцій та структуру програмних компонентів;
6. реалізувати програмний прототип інформаційної системи з підтримкою ролей користувачів, обліку завдань, контролю статусів і формування звітів;
7. провести тестування розробленої системи та оцінити ефективність її використання для підтримки дистанційного керування бізнес-процесами.

Об’єктом дослідження є процес дистанційного керування бізнес-процесами підприємства.

**Предметом дослідження** є методи, моделі та програмні засоби створення інформаційної системи для планування, контролю та аналізу діяльності бізнесу в дистанційному режимі.

**Наукова новизна роботи** полягає у розробленні підходу до побудови інформаційної системи, яка поєднує функції керування користувачами, обліку завдань, контролю виконання та формування аналітичної звітності в єдиному програмному середовищі. Запропоноване рішення забезпечує структуровану взаємодію між власником бізнесу, менеджером і співробітником, що дозволяє підвищити прозорість управлінських процесів та оперативність прийняття рішень.

**Методи дослідження** включають застосування системного аналізу для вивчення предметної області, UML-моделювання для опису функціональної та структурної організації системи, методи об’єктно-орієнтованого проєктування для побудови програмної логіки, а також методи реляційного моделювання даних для розроблення бази даних. Для реалізації інформаційної системи використовується клієнт-серверний підхід, механізми збереження даних у базі даних та програмні засоби оброблення управлінської інформації.

**Практична значущість одержаних результатів** полягає у можливості використання розробленої інформаційної системи в діяльності малого або середнього бізнесу для автоматизації процесів дистанційного управління. Система забезпечує централізований облік користувачів, створення та призначення завдань, контроль строків виконання, оновлення статусів, формування звітів і перегляд аналітичних показників. Використання такого програмного рішення дозволяє зменшити навантаження на керівників і менеджерів, підвищити дисципліну виконання завдань, покращити прозорість бізнес-процесів та забезпечити більш обґрунтоване прийняття управлінських рішень.

# 1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Опис предметної області

Дистанційне керування бізнесом є важливою складовою сучасного управління підприємством, оскільки значна частина організаційних, виконавчих та аналітичних процесів може здійснюватися без безпосередньої присутності керівника або менеджера на робочому місці. Такий підхід особливо актуальний для малого та середнього бізнесу, де власник або керівник одночасно контролює виконання завдань, роботу персоналу, поточні показники діяльності та формування звітності.

Предметна область інформаційної системи дистанційного керування бізнесом охоплює процеси планування роботи, призначення завдань виконавцям, контролю їх виконання, обліку статусів, перегляду ключових показників та формування управлінських звітів. Основними учасниками цього процесу є власник бізнесу, менеджер, співробітник та адміністратор системи. Кожен з них виконує окрему роль у межах інформаційної системи та має визначений набір функціональних можливостей.

Власник бізнесу здійснює загальний контроль діяльності підприємства, переглядає аналітичні показники, оцінює стан виконання робіт і приймає управлінські рішення. Менеджер відповідає за створення завдань, призначення виконавців, контроль строків і формування звітів. Співробітник отримує призначені завдання, виконує їх і оновлює поточний статус. Адміністратор забезпечує керування обліковими записами, ролями користувачів, довідниками та параметрами доступу.

У традиційних умовах такі процеси часто реалізуються за допомогою електронних таблиць, месенджерів, поштового листування або окремих сервісів для обліку задач. Це створює низку проблем: дублювання інформації, втрату актуальних даних, складність контролю відповідальних осіб, відсутність єдиної історії виконання завдань і недостатню прозорість управлінських процесів. Крім

того, керівник не завжди має можливість швидко отримати зведену інформацію про стан справ, кількість виконаних або прострочених завдань, ефективність роботи персоналу та загальну динаміку діяльності підприємства.

Інформаційна система дистанційного керування бізнесом призначена для усунення зазначених недоліків шляхом централізації даних і автоматизації основних управлінських операцій. Вона забезпечує єдине середовище для збереження користувачів, ролей, завдань, статусів, призначень, звітів і показників діяльності. Завдяки цьому всі учасники бізнес-процесу працюють з актуальною інформацією, а керівник отримує можливість оперативно оцінювати стан виконання робіт.

Загальну логіку взаємодії користувачів із системою подано на DFD-діаграмі нульового рівня, зображеній на рисунку 1.1. Діаграма відображає основні зовнішні сутності, центральний процес інформаційної системи та ключові сховища даних. Вона показує, що система приймає запити від користувачів, обробляє інформацію про завдання, ролі, звіти та показники, після чого повертає результати у вигляді дашбордів, статусів виконання, повідомлень і аналітичних звітів.

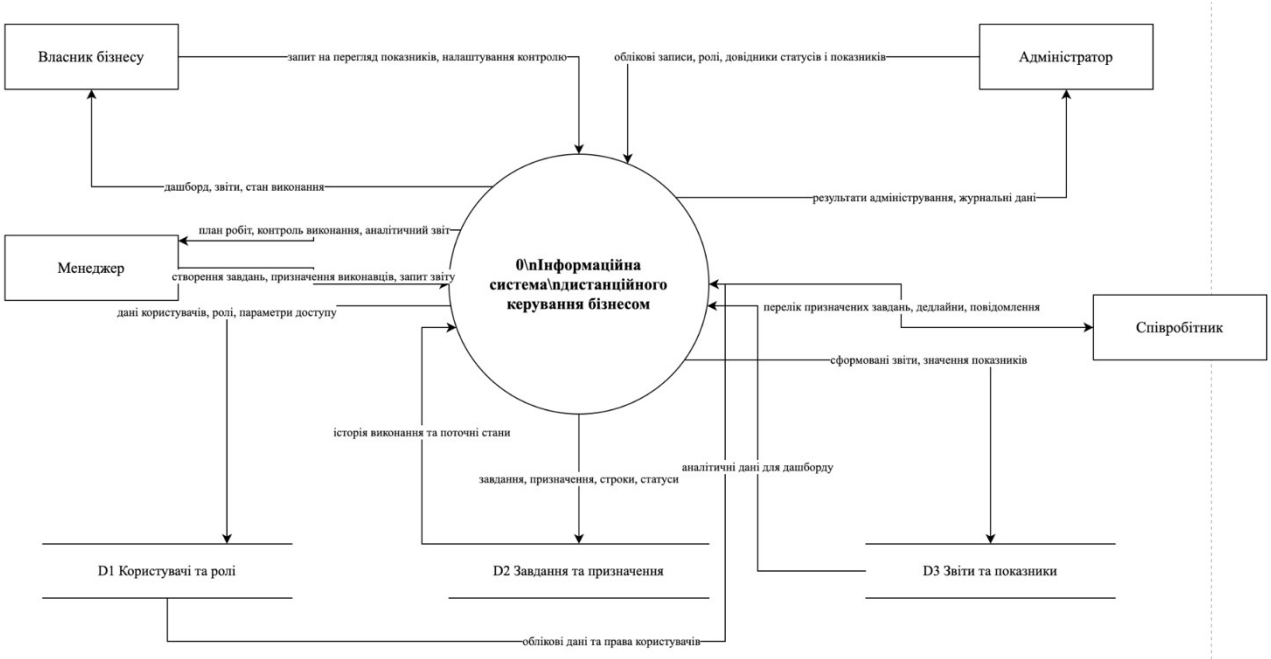


Рисунок 1.1 – DFD-діаграма нульового рівня інформаційної системи дистанційного керування бізнесом

Основні процеси предметної області наведено в таблиці 1.1.

Таблиця 1.1 – Основні процеси предметної області інформаційної системи дистанційного керування бізнесом

| № | Процес | Зміст процесу | Основний виконавець | Результат |
|---|---|---|---|---|
| 1 | Авторизація користувача | Перевірка облікових даних, визначення ролі та прав доступу | Власник бізнесу, менеджер, співробітник, адміністратор | Надання доступу до відповідних функцій системи |
| 2 | Керування користувачами та ролями | Створення, редагування й обмеження доступу користувачів | Адміністратор | Актуальна структура користувачів і прав доступу |
| 3 | Створення завдань | Внесення назви, опису, строку виконання, пріоритету та відповідального виконавця | Менеджер | Нове завдання в системі |
| 4 | Призначення та виконання завдань | Передавання завдання співробітнику, виконання роботи та оновлення статусу | Менеджер, співробітник | Актуальний стан виконання завдання |
| 5 | Контроль виконання | Перевірка строків, статусів, відповідальних осіб і прострочених завдань | Власник бізнесу, менеджер | Оцінка поточного стану робіт |
| 6 | Формування звітів | Узагальнення даних про завдання, виконання, прострочення та показники діяльності | Менеджер, власник бізнесу | Аналітичний звіт для прийняття рішень |
| 7 | Перегляд показників бізнесу | Відображення кількісних та якісних показників у вигляді дашборду | Власник бізнесу | Оперативна управлінська інформація |
| 8 | Журналювання дій | Фіксація важливих операцій користувачів у системі | Система, адміністратор | Контроль змін і підвищення прозорості роботи |

Основними інформаційними об’єктами предметної області є користувач, роль, завдання, призначення завдання, статус виконання, звіт та показник

діяльності. Користувач характеризується персональними даними, обліковими параметрами та роллю в системі. Роль визначає рівень доступу до функціональних можливостей. Завдання містить опис роботи, строк виконання, пріоритет і зв’язок із виконавцем. Статус завдання дозволяє відстежувати його поточний стан, наприклад: нове, у процесі виконання, виконано, скасовано або прострочено. Звіти та показники використовуються для узагальнення результатів роботи й аналітичної підтримки управління.

Таблиця 1.2 – Основні інформаційні об’єкти системи

| № | Інформаційний об’єкт | Призначення |
|---|---|---|
| 1 | Користувач | Зберігає дані про власника бізнесу, менеджера, співробітника або адміністратора |
| 2 | Роль | Визначає права доступу користувача до функцій системи |
| 3 | Завдання | Описує роботу, яку необхідно виконати в межах бізнес-процесу |
| 4 | Призначення завдання | Пов’язує завдання з конкретним виконавцем і відповідальним менеджером |
| 5 | Статус завдання | Відображає поточний етап виконання завдання |
| 6 | Звіт | Містить узагальнену інформацію про роботу за певний період |
| 7 | Показник діяльності | Використовується для оцінювання ефективності виконання завдань і роботи персоналу |
| 8 | Журнал дій | Фіксує важливі операції користувачів для контролю та аудиту |

З технічної точки зору предметна область передбачає роботу з даними, які постійно змінюються: створюються нові завдання, оновлюються статуси, змінюються відповідальні особи, формуються звіти та перераховуються аналітичні показники. Тому система має забезпечувати цілісність даних, розмежування доступу, швидке отримання інформації та можливість подальшого розширення функціоналу.

Важливою особливістю інформаційної системи є підтримка рольової моделі доступу. Для власника бізнесу пріоритетними є перегляд дашборду, звітів і загального стану виконання робіт. Для менеджера основними функціями є планування, створення завдань, призначення виконавців і контроль строків. Для

співробітника система повинна забезпечувати зручний перегляд власних завдань і можливість змінювати статус виконання. Адміністратор відповідає за підтримку працездатності системи, керування користувачами та налаштування довідкової інформації.

Отже, предметна область інформаційної системи дистанційного керування бізнесом охоплює сукупність управлінських, виконавчих та аналітичних процесів, пов’язаних із плануванням роботи підприємства, контролем завдань і формуванням звітності. Розроблення такої системи дозволяє підвищити прозорість бізнес-процесів, зменшити ручне опрацювання даних, забезпечити оперативний доступ до управлінської інформації та створити основу для прийняття обґрунтованих рішень.

## 1.2 Аналіз існуючих рішень

Для визначення функціональних можливостей розроблюваної інформаційної системи дистанційного керування бізнесом було проаналізовано сучасні програмні рішення, що використовуються для організації командної роботи, постановки завдань, контролю виконання, формування звітності та відстеження показників діяльності. Найближчими за призначенням до розроблюваної системи є Trello, Asana, ClickUp, monday.com та Bitrix24.

На рисунку 1.2 подано інтерфейс системи Trello. Це веборієнтований інструмент управління проєктами, який використовує дошки, списки та картки для організації задач. Trello добре підходить для візуального контролю простих робочих процесів, розподілу завдань між учасниками команди, встановлення строків і позначення пріоритетів. Перевагою системи є простота інтерфейсу та швидке освоєння користувачами. Водночас Trello має обмеження для повноцінного дистанційного керування бізнесом, оскільки його основний акцент зроблено на задачах і дошках, а не на комплексному обліку ролей, бізнес-показників, управлінських звітів і контролю ефективності персоналу.


Рисунок 1.2 – Інтерфейс системи Trello для керування завданнями

На рисунку 1.3 наведено приклад інтерфейсу Asana. Система призначена для управління командною роботою, проєктами, задачами, строками виконання та навантаженням працівників. Asana підтримує відстеження роботи від початку до завершення, дозволяє будувати робочі процеси, контролювати прогрес і пов’язувати проєкти з цілями організації. Недоліком для теми цієї роботи є те, що Asana є універсальною платформою для проєктного менеджменту, тому для невеликого бізнесу частина функцій може бути надлишковою, а спеціалізована модель “власник бізнесу - менеджер - співробітник” потребує додаткового налаштування.

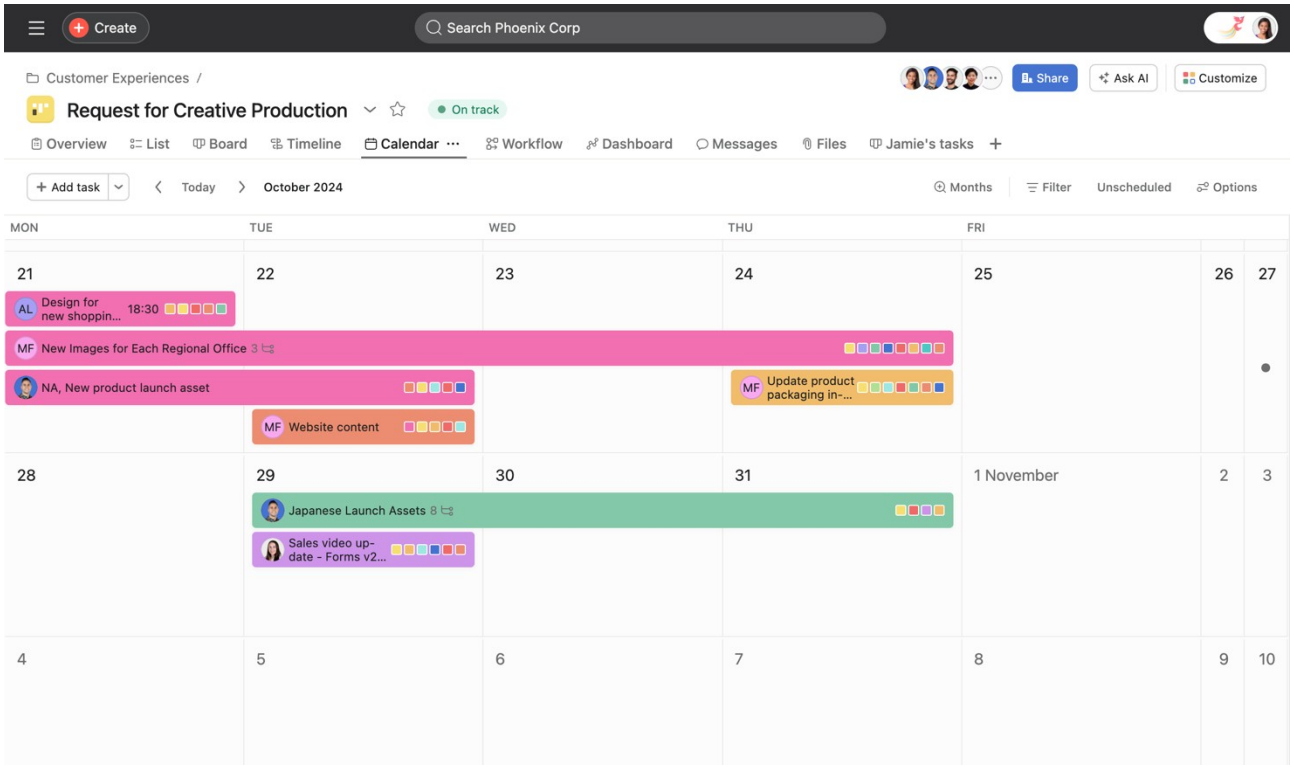


Рисунок 1.3 – Інтерфейс системи Asana для керування проєктами

На рисунку 1.4 представлено систему ClickUp. Вона поєднує інструменти для керування завданнями, проєктами, цілями, звітами та дашбордами. ClickUp дозволяє створювати різні подання для проєктів, фільтрувати завдання, контролювати прогрес і використовувати дашборди для відстеження цілей або OKR. Перевагою цього рішення є широкий набір функцій, проте саме через велику кількість можливостей система може бути складною для швидкого впровадження в невеликому бізнесі, де потрібен простий інструмент для постановки завдань, контролю виконання та формування звітів.

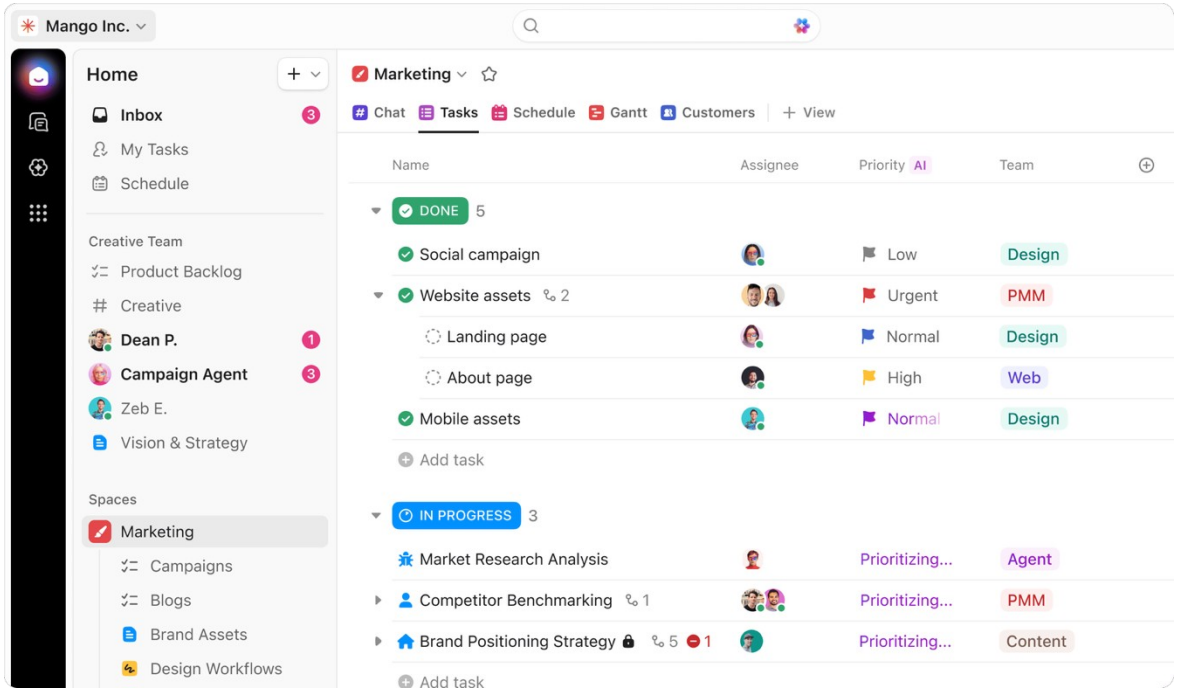


Рисунок 1.4 – Робочий простір ClickUp із завданнями та аналітичними панелями

На рисунку 1.5 показано інтерфейс monday.com. Платформа орієнтована на управління робочими процесами, проєктами, командами, продажами, маркетингом, IT та операційною діяльністю. monday.com має дашборди для відстеження прогресу, виявлення вузьких місць і прийняття рішень на основі даних. Для розроблюваної системи важливим є підхід monday.com до візуалізації стану робіт, однак для локального або навчального проєкту така платформа є занадто масштабною, а її використання залежить від тарифних планів і хмарної інфраструктури.

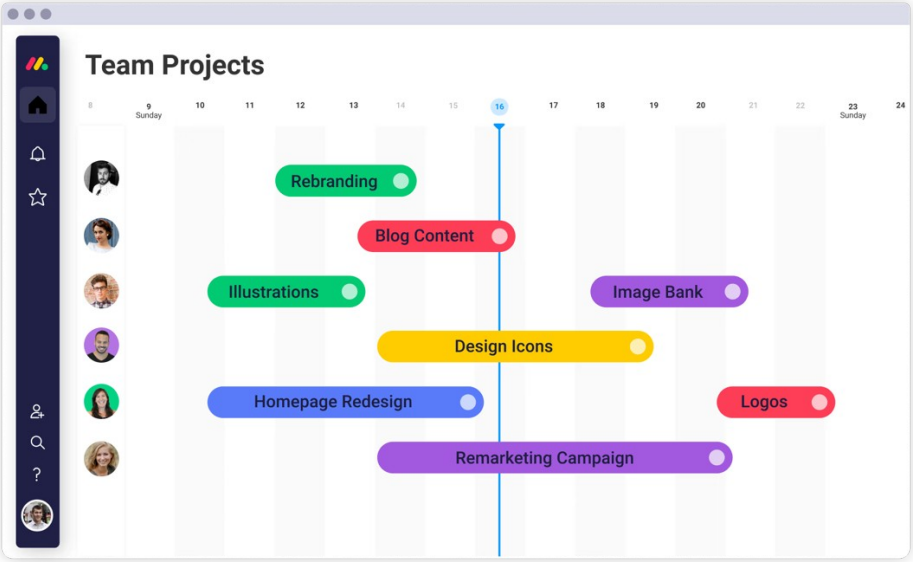


Рисунок 1.5 – Інтерфейс monday.com для контролю робочих процесів

На рисунку 1.6 наведено інтерфейс Bitrix24. Це комплексна бізнес-платформа, що включає CRM, задачі, проєкти, комунікації, звіти, контроль строків і KPI. Bitrix24 найближчий до теми дистанційного керування бізнесом, оскільки поєднує інструменти для організації роботи команди, управління клієнтами й контролю виконання завдань. Водночас система є багатофункціональною та складною для адаптації під вузький сценарій навчального проєкту, де необхідно реалізувати компактну систему з базовими ролями, завданнями, статусами та звітами.

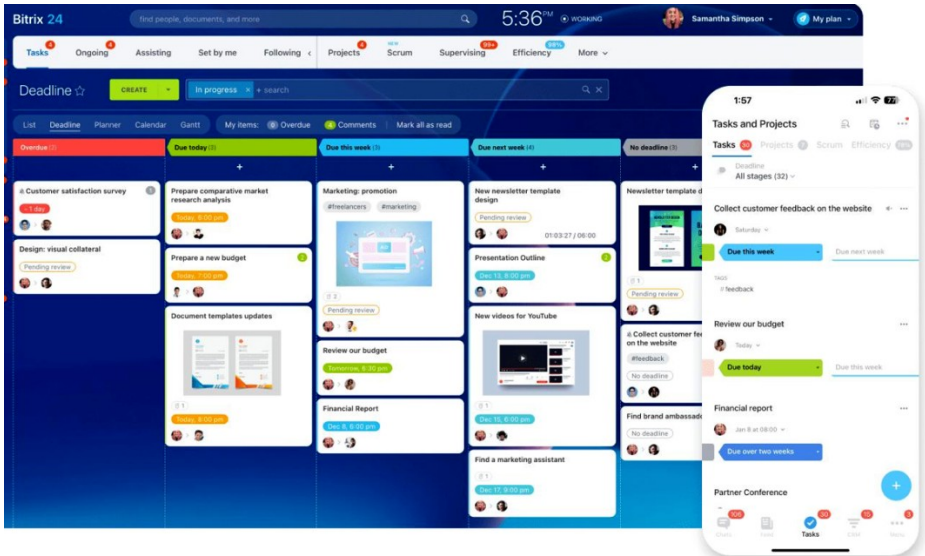


Рисунок 1.6 – Інтерфейс Bitrix24 для керування задачами та бізнес-процесами

Порівняльний аналіз існуючих рішень наведено в таблиці 1.2.

Таблиця 1.2 – Порівняння існуючих систем та розроблюваної інформаційної системи

| Система | Основне призначення | Контроль завдань | Аналітика та звіти | Переваги | Основні обмеження |
|---|---|---|---|---|---|
| Trello | Візуальне керування задачами | Так | Обмежено | Простий інтерфейс, дошки, картки, дедлайни | Немає повної моделі бізнес-контролю та звітності |
| Asana | Управління проєктами та командною роботою | Так | Так | Контроль строків, цілей, навантаження команди | Надлишкова функціональність для малого бізнесу |
| ClickUp | Комплексне керування задачами, цілями та дашбордами | Так | Так | Гнучкі подання, дашборди, фільтри, цілі | Складність налаштування через велику кількість модулів |
| monday.com | Управління робочими процесами підприємства | Так | Так | Візуальні дашборди, автоматизація, командна робота | Орієнтація на хмарне використання та тарифні плани |
| Bitrix24 | CRM, задачі, проєкти, комунікації, KPI | Так | Так | Багато бізнес-функцій в одній системі | Складність, перевантаженість інтерфейсу, надлишкові модулі |
| Розроблювана система | Дистанційне керування бізнесом | Так | Так | Орієнтація на ролі власника, менеджера й співробітника; проста структура; локальна БД | Обмежена функціональність порівняно з великими платформами |

Результати аналізу показують, що більшість існуючих систем добре вирішує окремі задачі: Trello зручний для простого візуального керування, Asana та ClickUp ефективні для проєктної роботи, monday.com підходить для

керування процесами, а Bitrix24 забезпечує ширший бізнес-функціонал. Проте ці рішення або мають надлишкову складність, або не орієнтовані безпосередньо на компактну модель дистанційного керування бізнесом із чітким розподілом ролей між власником, менеджером і співробітником.

Отже, доцільною є розробка власної інформаційної системи, яка буде зосереджена на базових процесах дистанційного керування бізнесом: авторизації користувачів, керуванні ролями, створенні та призначенні завдань, оновленні статусів, контролі строків, формуванні звітів і перегляді ключових показників. Такий підхід дозволяє уникнути перевантаження зайвими модулями та забезпечити просту, зрозумілу й адаптовану до потреб малого або середнього бізнесу систему.

### 1.3 Моделювання програмної системи

Моделювання предметної області інформаційної системи дистанційного керування бізнесом виконано для формалізації основних процесів, ролей користувачів та взаємодії між функціональними модулями системи. Основна увага приділена процесам авторизації, перегляду показників бізнесу, планування та призначення завдань, контролю виконання, формування звітів, перегляду власних завдань і оновлення статусу виконання.

У межах предметної області виділено три основні категорії користувачів: власник бізнесу, менеджер і співробітник. Власник бізнесу здійснює загальний контроль діяльності, переглядає показники та аналізує стан виконання робіт. Менеджер відповідає за планування завдань, призначення виконавців, контроль виконання і формування звітів. Співробітник працює з власними завданнями, виконує їх та оновлює поточний статус.

На рисунку 1.7 подано діаграму прецедентів інформаційної системи дистанційного керування бізнесом.

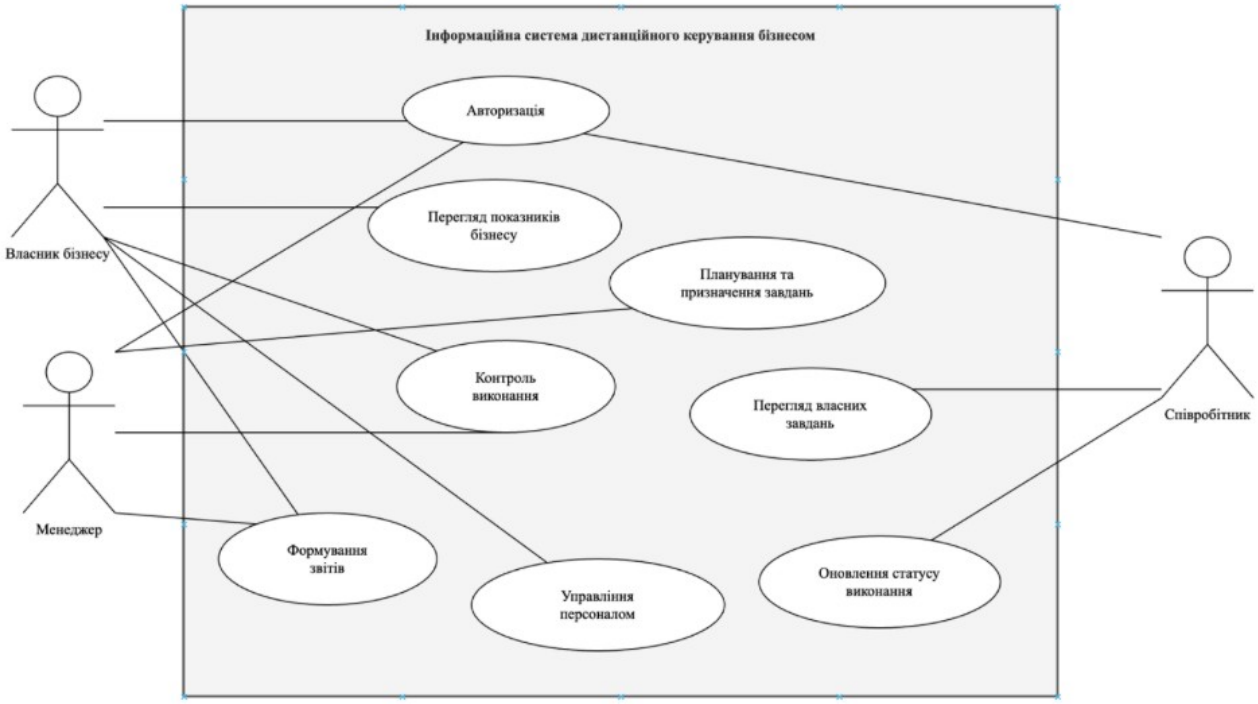


Рисунок 1.7 – Діаграма прецедентів інформаційної системи дистанційного керування бізнесом

Діаграма прецедентів відображає функціональні можливості системи з позиції користувачів. До основних прецедентів належать авторизація, перегляд показників бізнесу, планування та призначення завдань, контроль виконання, формування звітів, управління персоналом, перегляд власних завдань і оновлення статусу виконання. Такий набір функцій відповідає логіці дистанційного керування бізнесом, оскільки охоплює як управлінські дії, так і виконавчі операції.

Таблиця 1.3 – Характеристика акторів системи

| Актор | Основна роль у системі | Основні функції |
|---|---|---|
| Власник бізнесу | Загальний контроль діяльності підприємства | Авторизація, перегляд показників бізнесу, контроль виконання, управління персоналом |
| Менеджер | Організація роботи та контроль завдань | Авторизація, планування завдань, призначення виконавців, формування звітів, контроль виконання |
| Співробітник | Виконання призначених завдань | Авторизація, перегляд власних завдань, оновлення статусу виконання |

Взаємодію користувачів із системою деталізовано за допомогою діаграми послідовності. Вона дозволяє показати порядок виконання основних операцій, обмін повідомленнями між користувачами, інтерфейсом системи та базою даних.

На рисунку 1.8 наведено діаграму послідовності для основних сценаріїв роботи інформаційної системи.

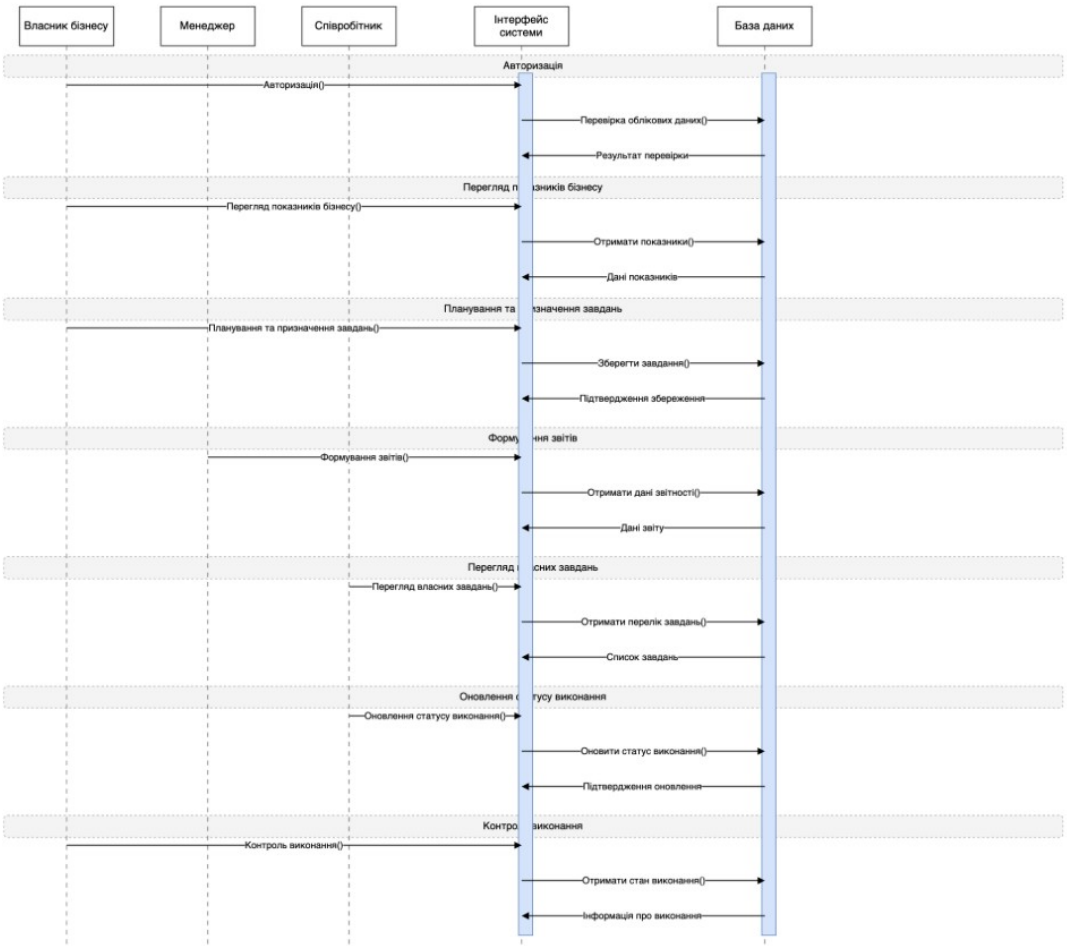


Рисунок 1.8 – Діаграма послідовності роботи інформаційної системи

Першим етапом взаємодії є авторизація користувача. Інтерфейс системи приймає облікові дані та передає їх до бази даних для перевірки. Після успішної перевірки користувач отримує доступ до функцій відповідно до своєї ролі. Власник бізнесу може переглядати показники діяльності та контролювати виконання робіт. Менеджер створює і призначає завдання, а також формує звіти на основі даних системи. Співробітник отримує перелік власних завдань і передає оновлений статус виконання.

Таблиця 1.4 – Основні сценарії взаємодії в системі

| № | Сценарій | Учасник | Вхідні дані | Результат |
|---|---|---|---|---|
| 1 | Авторизація | Власник бізнесу, менеджер, співробітник | Логін і пароль | Доступ до функцій системи |
| 2 | Перегляд показників бізнесу | Власник бізнесу | Запит на аналітичні дані | Відображення показників діяльності |
| 3 | Планування та призначення завдань | Менеджер | Назва, опис, строк, виконавець | Створене і призначене завдання |
| 4 | Перегляд власних завдань | Співробітник | Запит на список завдань | Перелік призначених завдань |
| 5 | Оновлення статусу виконання | Співробітник | Новий статус завдання | Актуалізований стан завдання |
| 6 | Контроль виконання | Власник бізнесу, менеджер | Запит на стан робіт | Інформація про виконання |
| 7 | Формування звіту | Менеджер | Період і параметри звіту | Сформований аналітичний звіт |

Для відображення загального алгоритму роботи системи побудовано діаграму активності. Вона описує логіку переходів між основними діями користувачів і системи, а також показує умови прийняття рішень під час виконання бізнес-процесу.

На рисунку 1.9 подано діаграму активності інформаційної системи дистанційного керування бізнесом.

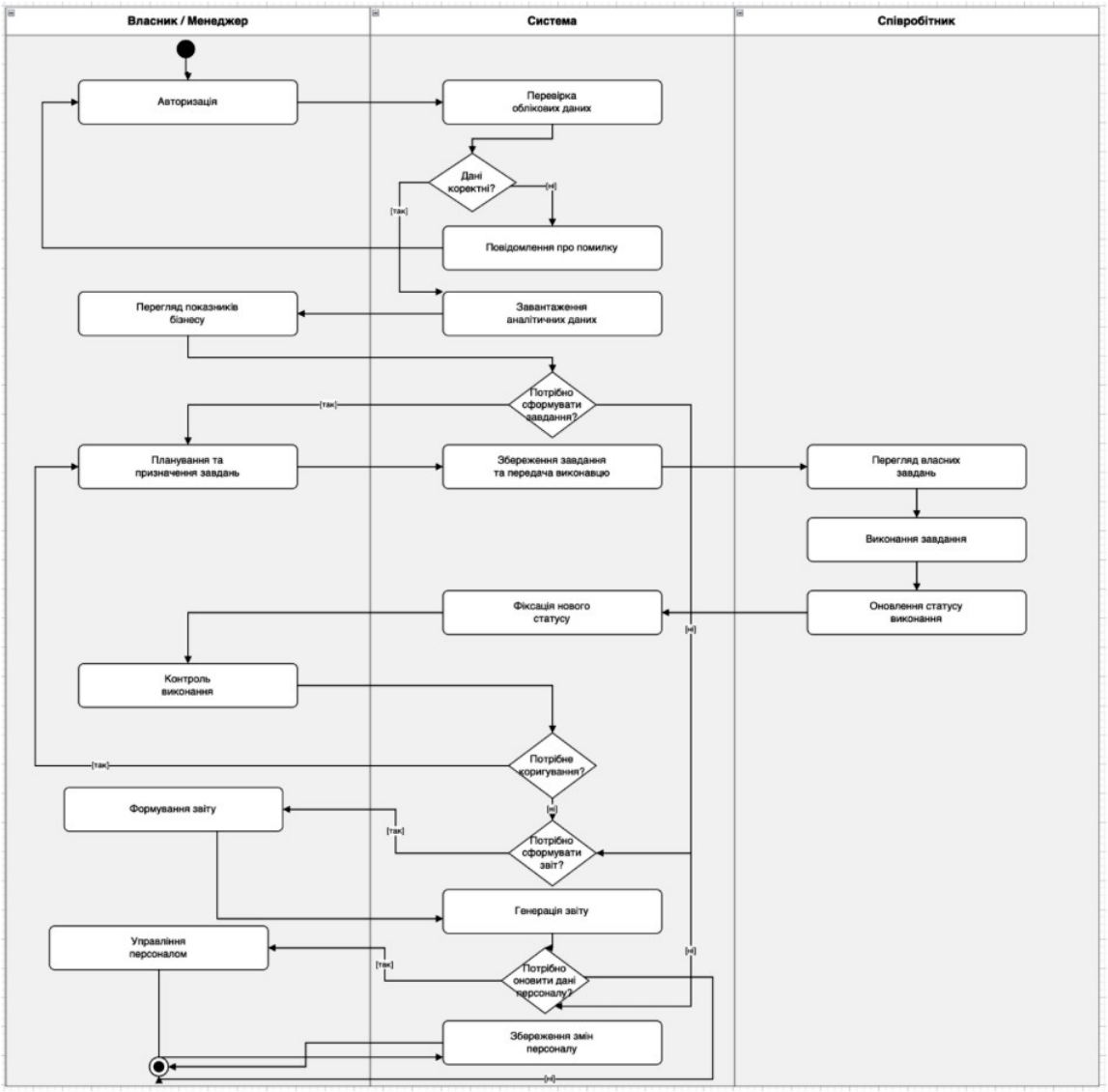


Рисунок 1.9 – Діаграма активності процесу дистанційного керування бізнесом

Процес роботи починається з авторизації користувача. Якщо введені дані некоректні, система формує повідомлення про помилку та повертає користувача до повторного введення облікових даних. Якщо перевірка успішна, система завантажує аналітичні дані та надає доступ до функцій відповідно до ролі користувача.

Менеджер або власник бізнесу може перейти до планування та призначення завдань. Після створення завдання система зберігає його в базі даних і передає виконавцю. Співробітник переглядає власні завдання, виконує роботу та оновлює статус. Новий статус фіксується системою, після чого

менеджер або власник бізнесу може виконати контроль. За потреби виконується коригування завдання, формування звіту або оновлення даних персоналу.

Таблиця 1.5 – Основні етапи процесу дистанційного керування бізнесом

| № | Етап | Зміст етапу | Результат |
|---|---|---|---|
| 1 | Авторизація | Перевірка облікових даних користувача | Вхід до системи або повідомлення про помилку |
| 2 | Завантаження даних | Отримання показників, завдань і статусів | Актуальні дані для роботи |
| 3 | Планування завдань | Створення завдання, визначення строків і виконавця | Нове завдання в системі |
| 4 | Передача завдання виконавцю | Збереження завдання та відображення його співробітнику | Завдання доступне для виконання |
| 5 | Виконання завдання | Робота співробітника із призначеним завданням | Зміна фактичного стану роботи |
| 6 | Оновлення статусу | Фіксація нового статусу виконання | Актуалізована інформація в системі |
| 7 | Контроль виконання | Аналіз строків, статусів і результатів | Виявлення виконаних або проблемних завдань |
| 8 | Формування звіту | Узагальнення даних за вибраний період | Аналітичний звіт для керівника |
| 9 | Управління персоналом | Оновлення даних користувачів і ролей | Актуальна структура персоналу |

Побудовані моделі дозволяють визначити основні вимоги до структури майбутньої інформаційної системи. Система повинна мати модуль авторизації, модуль керування користувачами, модуль завдань, модуль контролю виконання, модуль звітності та модуль аналітичних показників. Кожен модуль виконує окрему функцію, але всі вони працюють із єдиною базою даних.

Таблиця 1.6 – Основні функціональні модулі системи

| Модуль | Призначення |
|---|---|
| Модуль авторизації | Перевірка облікових даних і визначення ролі користувача |
| Модуль користувачів і ролей | Збереження даних персоналу, ролей і прав доступу |
| Модуль завдань | Створення, редагування та призначення завдань |
| Модуль виконання | Перегляд власних завдань і оновлення статусів |
| Модуль контролю | Перевірка строків, статусів і відповідальних осіб |
| Модуль звітності | Формування звітів за результатами виконання робіт |

Отже, моделювання предметної області дозволило формалізувати основні процеси дистанційного керування бізнесом, визначити ролі користувачів, функціональні сценарії та логіку роботи системи. Побудовані UML-діаграми є основою для подальшого проєктування бази даних, програмної архітектури та реалізації функціональних модулів інформаційної системи.

## 1.4 Аналіз вимог інформаційної системи

Аналіз вимог програмної системи виконано з метою визначення основних функцій, обмежень, технічних параметрів і вимог до безпеки інформаційної системи дистанційного керування бізнесом. Система повинна забезпечувати роботу кількох категорій користувачів, збереження даних про завдання, контроль виконання, формування звітів та перегляд показників діяльності підприємства.

Основними користувачами системи є власник бізнесу, менеджер, співробітник та адміністратор. Для кожної ролі передбачається окремий набір доступних функцій. Це дозволяє обмежити доступ до службових даних і забезпечити коректну організацію управлінських та виконавчих процесів.

Таблиця 1.7 – Вимоги до користувацьких ролей системи

| Роль користувача | Основні можливості | Обмеження доступу |
|---|---|---|
| Власник бізнесу | Перегляд показників, контроль виконання, перегляд звітів, аналіз роботи персоналу | Не виконує технічне адміністрування системи |
| Менеджер | Створення завдань, призначення виконавців, контроль строків, формування звітів | Не має доступу до системних налаштувань без дозволу адміністратора |
| Співробітник | Перегляд власних завдань, оновлення статусу виконання, додавання коментарів | Не може переглядати чужі завдання та змінювати ролі користувачів |
| Адміністратор | Керування користувачами, ролями, довідниками та параметрами доступу | Не відповідає за фактичне виконання бізнес-завдань |

Функціональні вимоги визначають перелік дій, які система повинна виконувати під час роботи користувачів. Вони охоплюють авторизацію, облік користувачів, керування завданнями, контроль виконання та формування звітності.

Таблиця 1.8 – Функціональні вимоги до інформаційної системи

| № | Вимога | Зміст вимоги |
|---|---|---|
| 1 | Авторизація користувачів | Система повинна забезпечувати вхід користувача за логіном і паролем |
| 2 | Розмежування прав доступу | Доступ до функцій має визначатися роллю користувача |
| 3 | Керування користувачами | Адміністратор повинен мати можливість створювати, редагувати та деактивувати облікові записи |
| 4 | Керування ролями | Система повинна підтримувати ролі власника бізнесу, менеджера, співробітника та адміністратора |
| 5 | Створення завдань | Менеджер повинен мати можливість створювати завдання з назвою, описом, строком і пріоритетом |
| 6 | Призначення виконавців | Система повинна дозволяти призначати завдання конкретному співробітнику |
| 7 | Перегляд власних завдань | Співробітник повинен бачити перелік завдань, які призначені саме йому |
| 8 | Оновлення статусу | Співробітник повинен мати можливість змінювати статус виконання завдання |
| 9 | Контроль виконання | Власник бізнесу та менеджер повинні бачити поточний стан виконання робіт |
| 10 | Формування звітів | Система повинна формувати звіти за вибраний період |
| 11 | Облік показників | Система повинна зберігати та відображати основні показники діяльності |
| 12 | Журналювання дій | Важливі операції користувачів мають фіксуватися для контролю змін |

До основних даних системи належать користувачі, ролі, завдання, статуси, призначення завдань, звіти та показники діяльності. Ці дані повинні зберігатися в єдиній базі даних, що забезпечує цілісність інформації та зв’язок між основними об’єктами системи.

Таблиця 1.9 – Вхідні та вихідні дані системи

| Тип даних | Дані | Джерело або одержувач |
|---|---|---|
| Вхідні дані | Логін, пароль, роль користувача | Користувач, адміністратор |

Продовження таблиці 1.9

| | | |
|---|---|---|
| Вхідні дані | Назва завдання, опис, строк, пріоритет, виконавець | Менеджер |
| Вхідні дані | Новий статус виконання, коментар до завдання | Співробітник |
| Вхідні дані | Період формування звіту, тип звіту | Менеджер, власник бізнесу |
| Вихідні дані | Список призначених завдань | Співробітник |
| Вихідні дані | Стан виконання робіт | Власник бізнесу, менеджер |
| Вихідні дані | Аналітичні показники | Власник бізнесу |
| Вихідні дані | Сформовані звіти | Власник бізнесу, менеджер |
| Вихідні дані | Повідомлення про помилки або успішне виконання операцій | Усі користувачі |

Нефункціональні вимоги описують якісні характеристики системи. Вони визначають швидкодію, надійність, зручність використання, масштабованість і підтримуваність програмного забезпечення.

Таблиця 1.10 – Нефункціональні вимоги до системи

| № | Вимога | Характеристика |
|---|---|---|
| 1 | Продуктивність | Основні операції мають виконуватися без помітної затримки для користувача |
| 2 | Надійність | Система повинна коректно обробляти помилкові дії користувачів |
| 3 | Зручність інтерфейсу | Інтерфейс має бути простим, зрозумілим і не перевантаженим зайвими елементами |
| 4 | Масштабованість | Структура системи має дозволяти додавання нових модулів і ролей |
| 5 | Підтримуваність | Код і структура бази даних мають бути логічно організовані для подальшого супроводу |
| 6 | Цілісність даних | Дані про користувачів, завдання, статуси та звіти повинні зберігатися узгоджено |
| 7 | Доступність | Система повинна бути доступною для користувачів у межах робочого середовища |
| 8 | Сумісність | Програмне забезпечення має працювати на типових робочих станціях користувачів |

Окрему увагу необхідно приділити вимогам до інформаційної безпеки. Оскільки система містить дані про користувачів, завдання, управлінські рішення та показники діяльності бізнесу, доступ до цієї інформації повинен бути контрольованим.

Таблиця 1.11 – Вимоги до інформаційної безпеки

| № | Вимога | Зміст вимоги |
|---|---|---|
| 1 | Ідентифікація користувача | Кожен користувач повинен працювати під власним обліковим записом |
| 2 | Аутентифікація | Вхід до системи має виконуватися після перевірки логіна і пароля |
| 3 | Авторизація | Доступ до функцій системи має визначатися роллю користувача |
| 4 | Захист паролів | Паролі користувачів повинні зберігатися у вигляді хешу |
| 5 | Обмеження доступу | Співробітник не повинен мати доступу до чужих завдань і адміністративних функцій |
| 6 | Контроль змін | Важливі операції мають фіксуватися в журналі дій |
| 7 | Перевірка введених даних | Система повинна перевіряти коректність форм, дат, статусів і обов’язкових полів |
| 8 | Захист від втрати даних | База даних повинна підтримувати резервне копіювання або відновлення інформації |

Технічні вимоги визначають мінімальні умови, необхідні для запуску та стабільної роботи програмного забезпечення. Оскільки система передбачає роботу з базою даних, її структура повинна підтримувати збереження користувачів, ролей, завдань, призначень, статусів, звітів і показників.

Таблиця 1.12 – Технічні вимоги до програмної системи

| Компонент | Вимога |
|---|---|
| Операційна система | Windows 10/11 або інша сумісна ОС |
| Середовище розробки | Visual Studio, VS Code або інше середовище для розробки програмного забезпечення |
| База даних | SQL Server або сумісна реляційна СУБД |
| Архітектура | Клієнт-серверна або модульна архітектура |
| Основні таблиці БД | users, roles, tasks, task_assignments, task_statuses, reports, indicators |
| Формат зберігання даних | Реляційна модель із первинними та зовнішніми ключами |
| Інтерфейс | Графічний або веб-орієнтований інтерфейс користувача |
| Підключення до БД | Через захищений рядок підключення або налаштований сервер бази даних |

З погляду користувача система повинна бути простою у використанні. Основні дії мають виконуватися через зрозумілі форми: вхід до системи, створення завдання, вибір виконавця, зміна статусу, перегляд звіту або показників. Для зменшення кількості помилок доцільно використовувати списки вибору для ролей, статусів, пріоритетів і виконавців.

З погляду системи важливо забезпечити коректну взаємодію між модулями. Під час створення завдання дані мають зберігатися в таблиці завдань, після чого формується запис про призначення виконавцю. Під час оновлення статусу система повинна змінювати стан призначення і фіксувати час редагування. Під час формування звіту система повинна підраховувати загальну кількість завдань, кількість виконаних, прострочених завдань і рівень виконання.

Отже, аналіз вимог показав, що розроблювана інформаційна система повинна забезпечувати повний цикл дистанційного керування бізнесом: від авторизації користувача та створення завдань до контролю виконання і формування аналітичної звітності. Визначені функціональні, нефункціональні, безпекові та технічні вимоги є основою для подальшого проєктування бази даних, архітектури системи та програмної реалізації.

### 1.5 Постановка завдання

На основі проведеного аналізу предметної області, сформульованих функціональних і нефункціональних вимог, а також результатів UML-моделювання було сформовано постановку завдання розроблення інформаційної системи підтримки надання освітніх послуг з повторного вивчення дисципліни студентами структурного підрозділу університету.

Завдання полягає у проєктуванні та розробленні інформаційної системи, призначеної для автоматизації процесів подання, розгляду, погодження та супроводу заяв на повторне вивчення дисциплін, забезпечення контролю

виконання регламентів та інтеграції з підсистемами навчального процесу і фінансового обліку.

Проєктована система розглядається як веб-орієнтований програмний комплекс із клієнт–серверною архітектурою, що функціонує у межах інформаційної інфраструктури університету. Система повинна забезпечувати повний життєвий цикл освітньої послуги з повторного вивчення дисципліни — від ініціювання студентом заявки до завершення повторного контролю та фіксації результатів у відповідних реєстрах.

Для досягнення поставленої мети необхідно спроєктувати архітектуру системи, що складається з взаємопов’язаних функціональних підсистем:

– підсистема користувацької взаємодії, яка забезпечує роботу студентів, методистів/деканату, бухгалтерії, викладачів і адміністратора через веб-інтерфейс;

– підсистема управління заявами, що відповідає за реєстрацію, перевірку, зміну статусів і зберігання заяв на повторне вивчення дисциплін;

– підсистема прийняття управлінських рішень, яка реалізує механізм розгляду та погодження заяв відповідно до встановлених регламентів;

– платіжна підсистема, що забезпечує формування рахунків та підтвердження оплати освітніх послуг (за наявності фінансової складової);

– підсистема інтеграції з навчальним процесом, яка забезпечує зарахування студента до групи повторного вивчення, формування доступів та фіксацію результатів повторного контролю.

Вхідними даними системи є:

– персональні та облікові дані користувачів (ідентифікатор, роль, параметри доступу);

– інформація про дисципліну (назва, семестр, форма контролю);

– дані про академічну заборгованість студента;

– параметри регламенту повторного вивчення (строки подання, кількість спроб, необхідність оплати);

– інформація про фінансові операції (рахунок, статус оплати).

Дані можуть надходити як безпосередньо від користувача через веб-інтерфейс, так і шляхом інтеграції з внутрішніми інформаційними системами університету (реєстр студентів, навчальні плани, підсистема фінансового обліку).

Вихідними даними системи є:

– статуси заяв на повторне вивчення (зареєстровано, на розгляді, схвалено, відхилено, зараховано);

– сформовані управлінські рішення та коментарі;

– інформація про необхідність та підтвердження оплати

– дані про зарахування студента до групи повторного вивчення;

– результати повторного контролю та їх фіксація у відповідних реєстрах;

– аналітичні звіти щодо кількості поданих заяв, прийнятих рішень та навантаження підрозділів.

Основною метою постановки завдання є створення інформаційної системи, яка забезпечує регламентовану, прозору та контрольовану процедуру надання освітньої послуги з повторного вивчення дисципліни. Автоматизація відповідних процесів дозволяє мінімізувати ризик помилок, скоротити час оброблення заяв, зменшити адміністративне навантаження на працівників структурного підрозділу та підвищити якість управління освітнім процесом.

Результатом виконання поставленого завдання має стати програмний продукт, який забезпечує централізований облік заяв, підтримку прийняття управлінських рішень, інтеграцію з фінансовими та навчальними підсистемами університету та формування єдиного інформаційного середовища для супроводу процесу повторного вивчення дисциплін.

# 2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних інформаційної системи дистанційного керування бізнесом розроблена для формалізації структури збереження інформації про користувачів, ролі, завдання, призначення виконавців, статуси виконання, звіти та аналітичні показники. Модель побудована у вигляді ER-діаграми та відображає основні сутності предметної області, їх атрибути, первинні ключі, зовнішні ключі та зв’язки між таблицями.

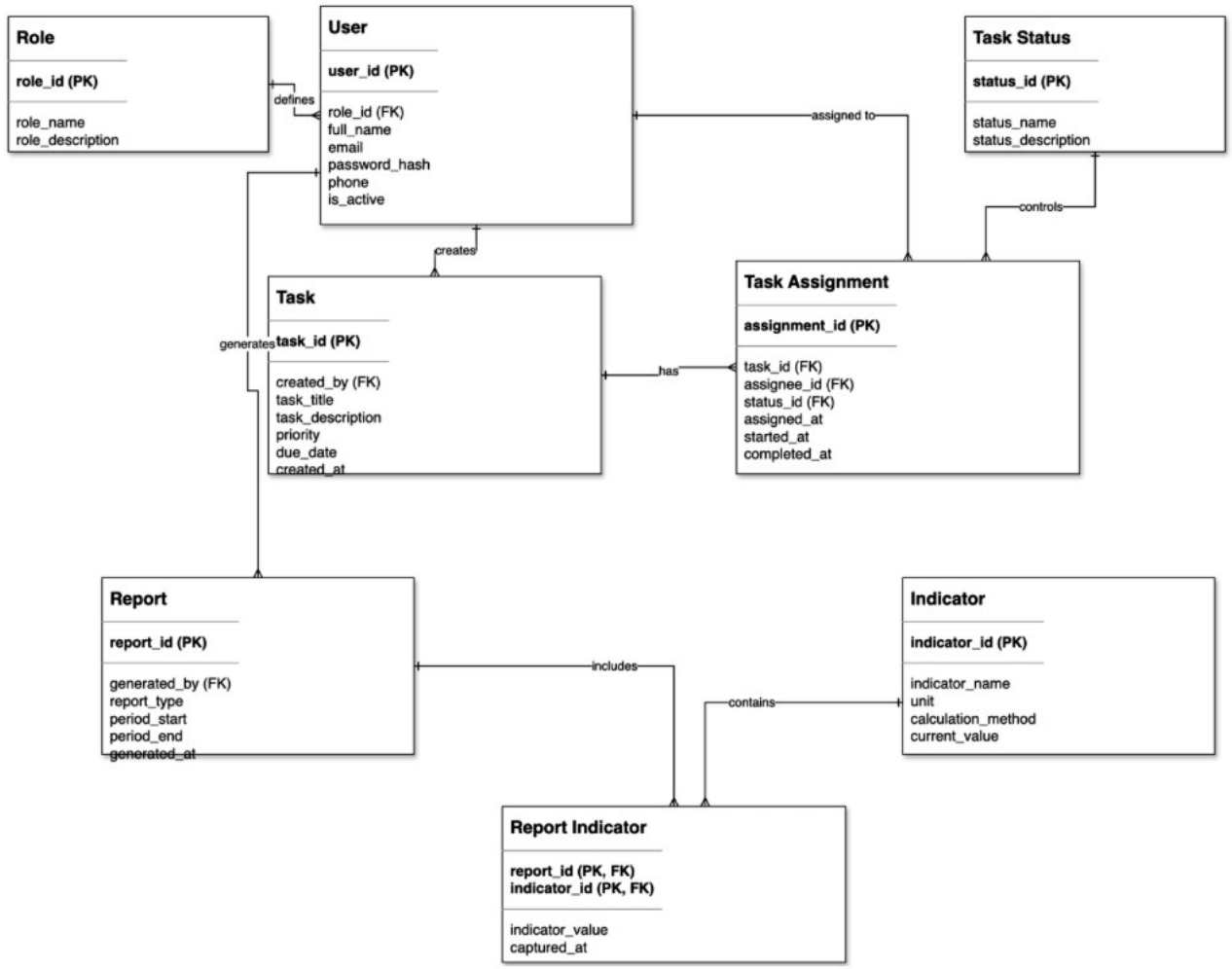


Рисунок 2.1 – Логічна модель даних інформаційної системи дистанційного керування бізнесом

До складу логічної моделі входять такі основні сутності: Role, User, Task, Task Assignment, Task Status, Report, Indicator та Report Indicator. Сутність Role використовується для збереження ролей користувачів системи. Вона пов’язана із сутністю User зв’язком один до багатьох, оскільки одна роль може бути призначена багатьом користувачам, але кожен користувач має одну основну роль.

Сутність User містить дані про користувачів системи: ідентифікатор, роль, ПІБ, електронну пошту, пароль у вигляді хешу, телефон та ознаку активності. Через зовнішній ключ role_id забезпечується зв’язок користувача з відповідною роллю. Такий підхід дає змогу реалізувати рольову модель доступу для власника бізнесу, менеджера, співробітника та адміністратора.

Сутність Task описує завдання, які створюються в системі. Вона містить назву, опис, пріоритет, дату виконання та дату створення. Поле created_by є зовнішнім ключем і вказує на користувача, який створив завдання. Це дозволяє зберігати інформацію про автора кожного завдання та забезпечувати контроль відповідальності.

Для фіксації факту призначення завдання конкретному виконавцю використано сутність Task Assignment. Вона пов’язує завдання, виконавця та статус виконання. Така структура дозволяє відокремити саме завдання від процесу його виконання. Завдяки цьому одне завдання може мати запис про призначення, виконавця, дату початку, дату завершення та поточний стан.

Сутність Task Status є довідником статусів виконання завдань. Вона містить назву статусу та його опис. Використання окремої таблиці статусів відповідає принципам нормалізації, оскільки повторювані значення не зберігаються безпосередньо в таблиці призначень, а задаються через зовнішній ключ status_id.

Сутність Report використовується для збереження сформованих звітів. Вона містить тип звіту, період, дату формування та користувача, який створив звіт. Звіти формуються на основі інформації про завдання, їх виконання та

статуси. Це забезпечує можливість аналізу роботи персоналу за визначений період.

Сутність Indicator зберігає перелік аналітичних показників, наприклад кількість завдань, кількість виконаних завдань, кількість прострочених завдань або рівень виконання. Для зв’язку звітів із показниками використано проміжну сутність Report Indicator. Вона реалізує зв’язок багато до багатьох між Report та Indicator, оскільки один звіт може містити кілька показників, а один показник може використовуватися в різних звітах.

Логічна модель даних приведена до третьої нормальної форми. Перша нормальна форма забезпечується тим, що всі атрибути мають атомарні значення, а повторювані групи даних відсутні. Друга нормальна форма дотримана завдяки тому, що неключові атрибути залежать від усього первинного ключа відповідної сутності. Третя нормальна форма забезпечується відсутністю транзитивних залежностей між неключовими атрибутами. Наприклад, дані про роль користувача винесені в окрему таблицю Role, статуси завдань зберігаються в Task Status, а показники звітів винесені в Indicator.

Нормалізація структури даних дозволяє уникнути дублювання інформації, зменшити ризик логічних помилок і забезпечити цілісність бази даних. Первинні ключі використовуються для однозначної ідентифікації записів, а зовнішні ключі забезпечують зв’язки між сутностями. Це дає змогу коректно зберігати дані про користувачів, завдання, призначення, статуси, звіти та показники діяльності.

Отже, побудована ER-діаграма відображає логічну структуру даних інформаційної системи дистанційного керування бізнесом. Вона є основою для подальшого створення фізичної моделі бази даних, визначення таблиць, типів даних, обмежень цілісності та SQL-скриптів реалізації.

### 2.2 Фізична модель даних розроблювальної системи

Фізична модель даних інформаційної системи дистанційного керування бізнесом побудована на основі логічної ER-моделі та відображає структуру

таблиць, типи даних, первинні й зовнішні ключі, а також основні обмеження цілісності. Модель орієнтована на реалізацію в реляційній СУБД PostgreSQL, оскільки в ній використано типи BIGSERIAL, SMALLSERIAL, TIMESTAMP, NUMERIC та обмеження NOT NULL, UNIQUE, PRIMARY KEY і FOREIGN KEY.

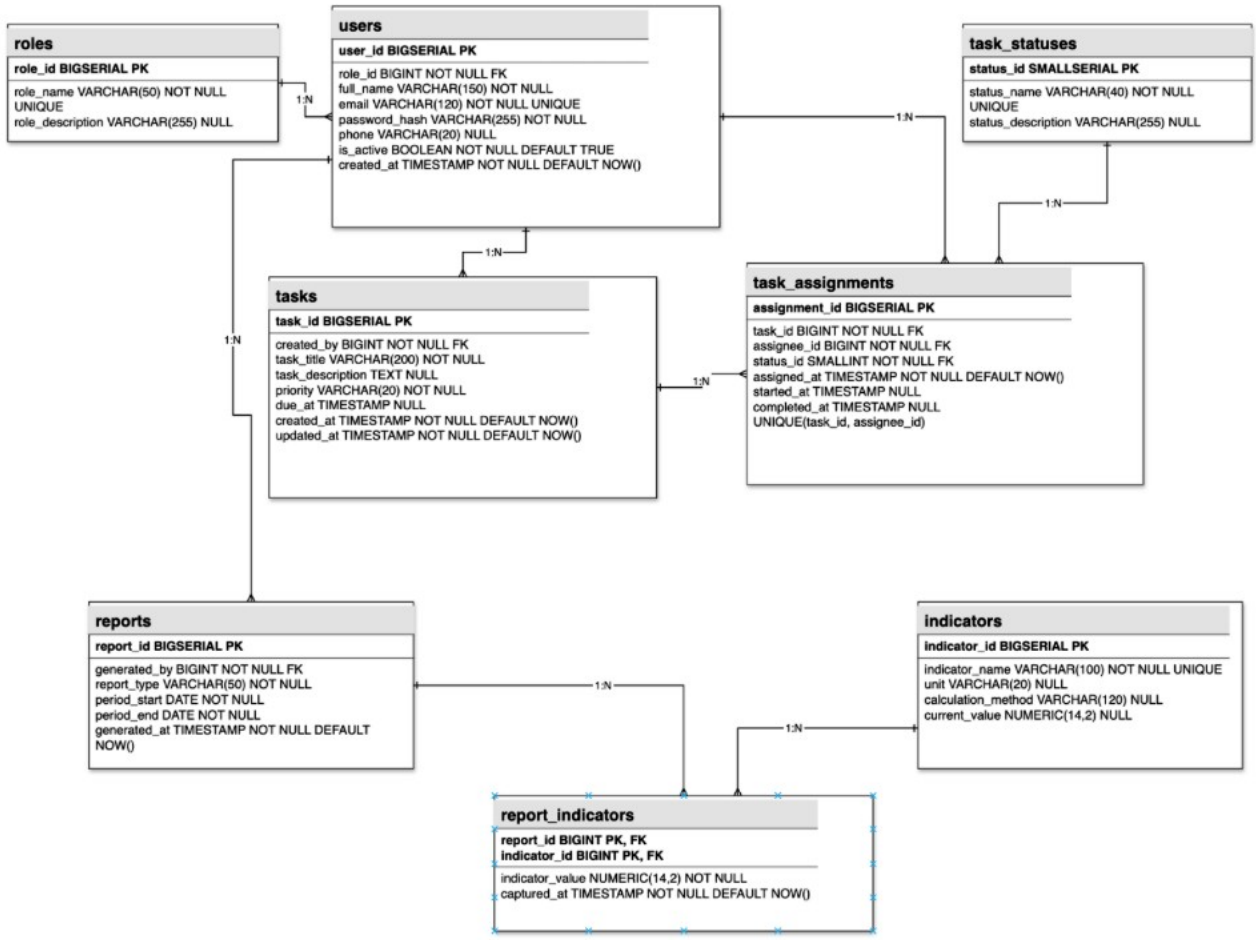


Рисунок 2.2 – Фізична модель даних інформаційної системи дистанційного керування бізнесом

До складу фізичної моделі входять таблиці roles, users, tasks, task_statuses, task_assignments, reports, indicators та report_indicators. Таблиці roles, task_statuses та indicators виконують роль довідників. Основні операційні дані зберігаються в таблицях users, tasks, task_assignments і reports. Для зв’язку звітів з аналітичними показниками використано проміжну таблицю report_indicators.

Модель відповідає вимогам третьої нормальної форми. Дані про ролі користувачів, статуси завдань і показники винесені в окремі таблиці, тому в

основних таблицях не дублюються текстові значення. Таблиця task_assignments відокремлює саме завдання від процесу його виконання, що дає змогу зберігати виконавця, статус, дату початку та дату завершення. Таблиця report_indicators реалізує зв’язок між звітами й показниками, тому один звіт може містити кілька значень аналітики.

Фрагмент SQL-коду створення довідкових таблиць наведено нижче.

```
CREATE TABLE roles (
    role_id BIGSERIAL PRIMARY KEY,
    role_name VARCHAR(50) NOT NULL UNIQUE,
    role_description VARCHAR(255)
);

CREATE TABLE task_statuses (
    status_id SMALLSERIAL PRIMARY KEY,
    status_name VARCHAR(40) NOT NULL UNIQUE,
    status_description VARCHAR(255)
);

CREATE TABLE indicators (
    indicator_id BIGSERIAL PRIMARY KEY,
    indicator_name VARCHAR(100) NOT NULL UNIQUE,
    unit VARCHAR(20),
    calculation_method VARCHAR(120),
    current_value NUMERIC(14,2)
);
```

Основна таблиця users зберігає облікові записи користувачів системи. Для кожного користувача задається роль, ПІБ, електронна пошта, хеш пароля, телефон, ознака активності та дата створення запису.

```
CREATE TABLE users (
    user_id BIGSERIAL PRIMARY KEY,
    role_id BIGINT NOT NULL REFERENCES roles(role_id),
    full_name VARCHAR(150) NOT NULL,
    email VARCHAR(120) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    phone VARCHAR(20),
    is_active BOOLEAN NOT NULL DEFAULT TRUE,
    created_at TIMESTAMP NOT NULL DEFAULT NOW()
);
```

Для збереження завдань використано таблицю tasks. Вона містить автора завдання, назву, опис, пріоритет, строк виконання та службові часові поля.

```
CREATE TABLE tasks (
    task_id BIGSERIAL PRIMARY KEY,
    created_by BIGINT NOT NULL REFERENCES users(user_id),
    task_title VARCHAR(200) NOT NULL,
    task_description TEXT,
    priority VARCHAR(20) NOT NULL,
    due_at TIMESTAMP,
    created_at TIMESTAMP NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMP NOT NULL DEFAULT NOW()
);
```

Призначення завдань виконавцям зберігається в таблиці task_assignments. Вона пов’язує завдання, користувача-виконавця та поточний статус виконання.

```
CREATE TABLE task_assignments (
    assignment_id BIGSERIAL PRIMARY KEY,
    task_id BIGINT NOT NULL REFERENCES tasks(task_id),
    assignee_id BIGINT NOT NULL REFERENCES users(user_id),
    status_id SMALLINT NOT NULL REFERENCES task_statuses(status_id),
    assigned_at TIMESTAMP NOT NULL DEFAULT NOW(),
    started_at TIMESTAMP,
    completed_at TIMESTAMP,
    UNIQUE(task_id, assignee_id)
);
```

Для формування управлінської звітності використано таблиці reports та report_indicators. Перша зберігає загальну інформацію про звіт, друга фіксує значення конкретних показників у межах звіту.

```
CREATE TABLE reports (
    report_id BIGSERIAL PRIMARY KEY,
    generated_by BIGINT NOT NULL REFERENCES users(user_id),
    report_type VARCHAR(50) NOT NULL,
    period_start DATE NOT NULL,
    period_end DATE NOT NULL,
    generated_at TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE TABLE report_indicators (
    report_id BIGINT NOT NULL REFERENCES reports(report_id),
    indicator_id BIGINT NOT NULL REFERENCES indicators(indicator_id),
    indicator_value NUMERIC(14,2) NOT NULL,
    captured_at TIMESTAMP NOT NULL DEFAULT NOW(),
    PRIMARY KEY (report_id, indicator_id)
);
```

Для заповнення довідників системи використовуються умовно-постійні дані. До них належать ролі користувачів, статуси виконання завдань і базові аналітичні показники.

```
INSERT INTO roles (role_name, role_description) VALUES
('Власник бізнесу', 'Користувач, який контролює діяльність підприємства'),
('Менеджер', 'Користувач, який створює завдання та формує звіти'),
('Співробітник', 'Користувач, який виконує призначені завдання'),
('Адміністратор', 'Користувач, який керує обліковими записами та довідниками');

INSERT INTO task_statuses (status_name, status_description) VALUES
('Нове', 'Завдання створено, але ще не розпочато'),
('У процесі', 'Завдання перебуває на виконанні'),
('Виконано', 'Завдання завершено виконавцем'),
('Скасовано', 'Завдання скасовано уповноваженим користувачем'),
('Прострочено', 'Строк виконання завдання порушено');

INSERT INTO indicators (indicator_name, unit, calculation_method) VALUES
('Кількість завдань', 'шт.', 'Підрахунок усіх завдань за період'),
('Кількість виконаних завдань', 'шт.', 'Підрахунок завдань зі статусом
Виконано'),
('Кількість прострочених завдань', 'шт.', 'Підрахунок завдань із порушеним
строком'),
```

```
('Рівень виконання', '%', 'Відношення виконаних завдань до загальної
кількості');
```

Для прискорення пошуку даних доцільно створити індекси за полями, які часто використовуються у фільтрації та зв’язках між таблицями.

```
CREATE INDEX idx_users_role_id ON users(role_id);
CREATE INDEX idx_tasks_created_by ON tasks(created_by);
CREATE INDEX idx_tasks_due_at ON tasks(due_at);
CREATE INDEX idx_assignments_task_id ON task_assignments(task_id);
CREATE INDEX idx_assignments_assignee_id ON task_assignments(assignee_id);
CREATE INDEX idx_assignments_status_id ON task_assignments(status_id);
CREATE INDEX idx_reports_generated_by ON reports(generated_by);
```

Отже, фізична модель даних забезпечує збереження основних об’єктів інформаційної системи дистанційного керування бізнесом і підтримує зв’язки між користувачами, завданнями, статусами, звітами та показниками. Використання первинних і зовнішніх ключів, унікальних обмежень та нормалізованої структури дозволяє зменшити дублювання даних і забезпечити коректну роботу програмних модулів.

### 2.3 Обґрунтування вибору технологічного стеку програмної системи

Для реалізації інформаційної системи дистанційного керування бізнесом обрано технологічний стек, який забезпечує швидку розробку, простий запуск, роботу з локальною базою даних та створення зручного графічного інтерфейсу. Оскільки система призначена для демонстрації основних процесів керування користувачами, завданнями, статусами, звітами та аналітичними показниками, доцільним є використання настільного застосунку на основі Python, PyQt6 і SQLite.

Основною мовою програмування обрано Python. Вона дає змогу швидко реалізувати бізнес-логіку системи, оброблення даних, авторизацію користувачів, фільтрацію завдань, формування звітів та експорт інформації. Python також добре підходить для навчальних і прикладних інформаційних систем, оскільки має зрозумілий синтаксис і підтримує модульну структуру програми.

Для створення графічного інтерфейсу використано PyQt6. Ця бібліотека дозволяє розробити повноцінний desktop-застосунок із вікнами, формами,

таблицями, кнопками, вкладками, діалоговими повідомленнями та панелями керування. У межах розроблюваної системи PyQt6 забезпечує створення окремих робочих областей для власника бізнесу, менеджера, співробітника та адміністратора.

Для збереження даних обрано SQLite. Її перевагою є те, що база даних зберігається у локальному файлі та не потребує встановлення окремого сервера. Це спрощує запуск програми, тестування та демонстрацію функціоналу. У базі даних зберігаються користувачі, ролі, завдання, призначення, статуси, звіти, показники, повідомлення та журнал дій.

Таблиця 2.1 – Обґрунтування вибору технологічного стеку програмної системи

| Компонент | Обрана технологія | Призначення у системі | Обґрунтування вибору |
|---|---|---|---|
| Мова програмування | Python | Реалізація бізнес-логіки, оброблення даних, робота з БД | Простий синтаксис, швидка розробка, зручність для прикладних систем |
| Графічний інтерфейс | PyQt6 | Побудова вікон, форм, таблиць, вкладок і діалогів | Дозволяє створити повноцінний desktop-застосунок із сучасним інтерфейсом |
| База даних | SQLite | Збереження користувачів, завдань, статусів, звітів і журналу дій | Не потребує окремого сервера, автоматично створюється у вигляді локального файлу |
| Стилізація інтерфейсу | QSS | Оформлення кнопок, панелей, таблиць і шрифтів | Забезпечує єдиний стиль програми та покращує читабельність інтерфейсу |
| Архітектура | Модульна desktop-архітектура | Розділення інтерфейсу, авторизації, БД і бізнес-логіки | Спрощує супровід програми та подальше розширення функціоналу |
| Авторизація | Хешування паролів | Захист облікових записів користувачів | Паролі не зберігаються у відкритому вигляді |
| Формат експорту | CSV | Вивантаження списку завдань і результатів роботи | Підтримується табличними редакторами та зручний для звітності |
| Середовище запуску | Windows, macOS | Запуск програми на робочих комп’ютерах користувачів | Дозволяє перевірити систему без складного серверного налаштування |

Структура програмного забезпечення побудована за модульним принципом. Основний файл main.py відповідає за запуск програми, побудову інтерфейсу та взаємодію користувача з функціями системи. Модуль database.pyзабезпечує створення таблиць, початкове заповнення бази даних і виконання SQL-запитів. Модуль auth.pyвикористовується для хешування паролів і перевірки облікових даних користувачів. Файли run_windows.bat і run_mac.command спрощують запуск застосунку на різних операційних системах.

У системі реалізовано рольову модель доступу. Після авторизації користувач отримує лише той набір функцій, який відповідає його ролі. Власник бізнесу працює з дашбордом, показниками та загальним контролем. Менеджер створює завдання, призначає виконавців і формує звіти. Співробітник переглядає власні завдання та оновлює їх статус. Адміністратор керує користувачами, довідниками та переглядає журнал дій.

Для оформлення інтерфейсу використано QSS-стилі. Це дозволило зробити інтерфейс більш читабельним, виділити інформаційні блоки, покращити вигляд таблиць і кнопок, а також створити єдиний стиль для всіх вікон програми. Такий підхід є важливим для системи дистанційного керування бізнесом, оскільки користувачі з різними ролями повинні швидко орієнтуватися у своїх функціях.

Отже, обраний стек Python, PyQt6 і SQLite є доцільним для реалізації інформаційної системи дистанційного керування бізнесом. Він забезпечує простоту запуску, локальне збереження даних, підтримку рольової моделі, зручний графічний інтерфейс і можливість подальшого розширення програмної системи.

## 2.4 Архітектура системи

Архітектура інформаційної системи дистанційного керування бізнесом побудована за багаторівневим принципом. Основними рівнями є клієнтський рівень, рівень прикладної логіки та рівень бази даних. Такий поділ дозволяє відокремити інтерфейс користувача від бізнес-логіки та збереження даних.

На рисунку 2.3 подано архітектуру інформаційної системи дистанційного керування бізнесом.

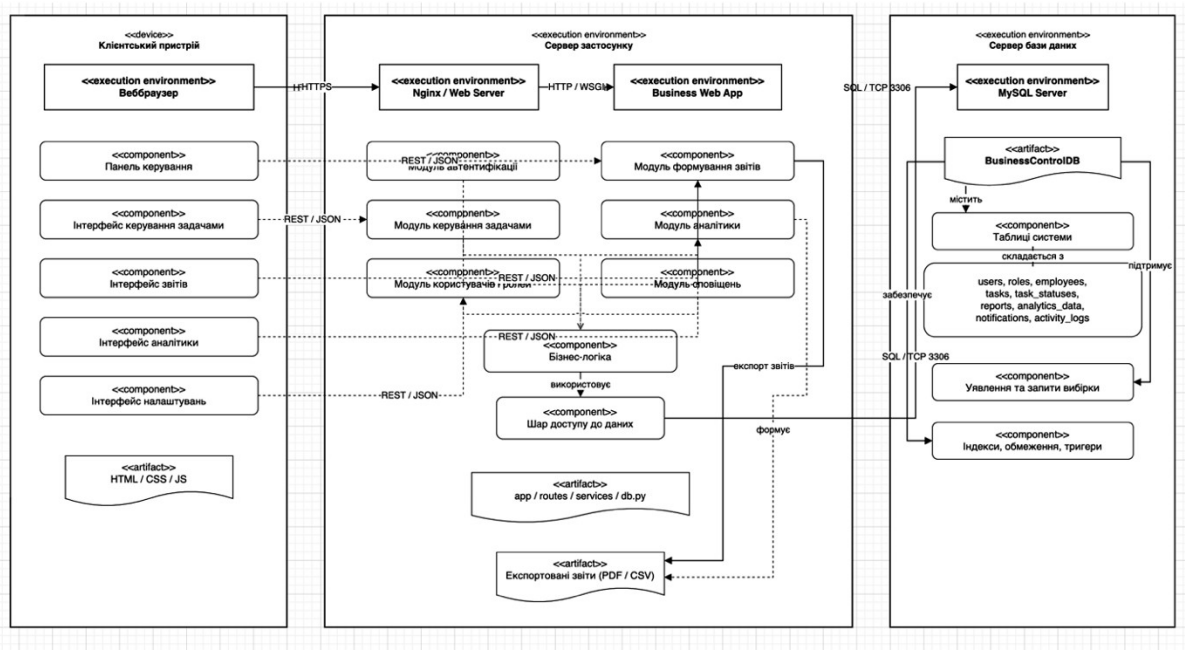


Рисунок 2.3 – Архітектура інформаційної системи дистанційного керування бізнесом

Клієнтський рівень забезпечує взаємодію користувача із системою. До нього належать панель керування, інтерфейс роботи із завданнями, інтерфейс звітів, аналітики та налаштувань. Через ці елементи власник бізнесу, менеджер, співробітник і адміністратор виконують дії відповідно до своїх прав доступу.

Рівень прикладної логіки відповідає за оброблення запитів користувачів. Він містить модуль автентифікації, модуль керування задачами, модуль користувачів і ролей, модуль формування звітів, модуль аналітики, модуль сповіщень, бізнес-логіку та шар доступу до даних. Саме на цьому рівні виконується перевірка прав користувача, створення завдань, зміна статусів, формування звітів і розрахунок показників.

Рівень бази даних призначений для збереження основної інформації системи. У базі містяться таблиці користувачів, ролей, завдань, статусів, звітів, показників, повідомлень і журналу дій. Для підтримки цілісності даних використовуються первинні ключі, зовнішні ключі, індекси, обмеження та запити вибірки.

Таблиця 2.2 – Основні компоненти архітектури системи

| Рівень системи | Компонент | Призначення |
|---|---|---|
| Клієнтський рівень | Панель керування | Відображення основної інформації та швидкий доступ до функцій |
| Клієнтський рівень | Інтерфейс керування задачами | Створення, перегляд і редагування завдань |
| Клієнтський рівень | Інтерфейс звітів | Перегляд сформованих звітів |
| Клієнтський рівень | Інтерфейс аналітики | Відображення ключових показників бізнесу |
| Клієнтський рівень | Інтерфейс налаштувань | Керування параметрами системи |
| Прикладний рівень | Модуль автентифікації | Перевірка облікових даних і визначення ролі користувача |
| Прикладний рівень | Модуль керування задачами | Створення, призначення та оновлення завдань |
| Прикладний рівень | Модуль користувачів і ролей | Адміністрування облікових записів і прав доступу |
| Прикладний рівень | Модуль звітів | Формування звітів за вибраний період |
| Прикладний рівень | Модуль аналітики | Розрахунок показників виконання |
| Прикладний рівень | Шар доступу до даних | Виконання запитів до бази даних |
| Рівень даних | База даних | Збереження користувачів, завдань, статусів, звітів і показників |

У межах реалізованого програмного прототипу архітектура адаптована під desktop-застосунок на PyQt6. Користувацький інтерфейс, бізнес-логіка та доступ до бази даних реалізовані в одному програмному середовищі, але логічно розділені на окремі модулі. Для збереження даних використовується локальна база SQLite, яка створюється автоматично під час запуску програми.

Основними файлами програмної реалізації є `main.py`, `database.py` та `auth.py`. Файл `main.py` відповідає за інтерфейс і

взаємодію користувача з програмою. Файл database.py забезпечує створення таблиць, початкове заповнення бази та виконання SQL-запитів. Файл auth.py використовується для хешування паролів і перевірки облікових даних.

Обрана архітектура забезпечує розмежування функцій між ролями користувачів. Власник бізнесу працює з дашбордом, показниками та контролем виконання. Менеджер створює завдання, призначає виконавців і формує звіти. Співробітник переглядає власні завдання та оновлює їх статус. Адміністратор керує користувачами, ролями, довідниками та журналом дій.

Отже, архітектура системи забезпечує структуровану організацію програмного забезпечення, підтримує рольову модель доступу, централізоване збереження даних, контроль виконання завдань і формування аналітичної звітності. Такий підхід є достатнім для реалізації інформаційної системи дистанційного керування бізнесом і дозволяє розширювати її функціональність у майбутньому.

### 2.5 Діаграма пакетів

Діаграма пакетів використовується для подання загальної програмної структури інформаційної системи дистанційного керування бізнесом. Вона показує поділ системи на логічні частини, їх призначення та залежності між клієнтською і серверною частинами програмного забезпечення.

На рисунку 2.4 подано діаграму пакетів інформаційної системи дистанційного керування бізнесом.

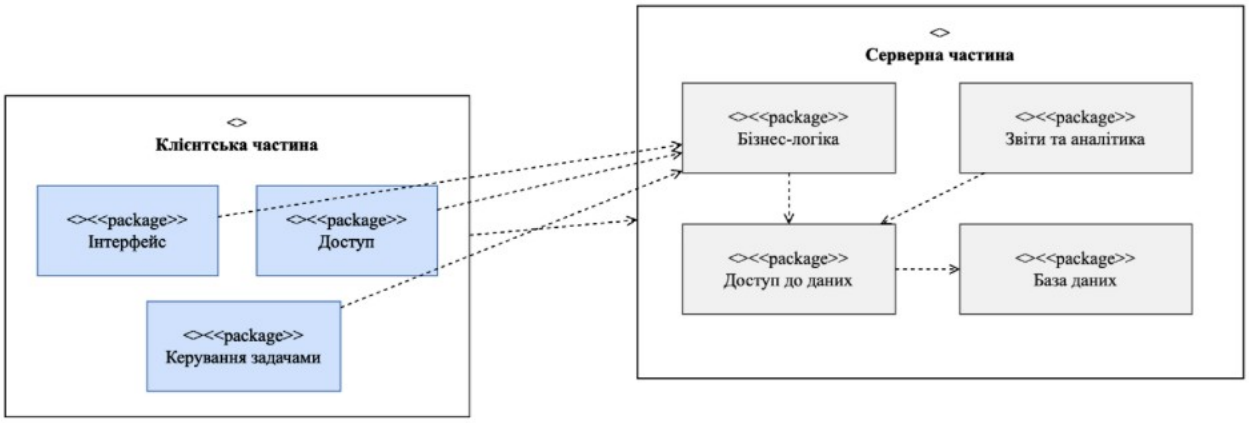


Рисунок 2.4 – Діаграма пакетів інформаційної системи дистанційного керування бізнесом

У клієнтській частині виділено пакети «Інтерфейс», «Доступ» та «Керування задачами». Пакет «Інтерфейс» відповідає за відображення вікон, форм, таблиць, кнопок і панелей користувача. Пакет «Доступ» забезпечує авторизацію користувача та перевірку його ролі. Пакет «Керування задачами» реалізує дії, пов’язані з переглядом, створенням, призначенням і оновленням статусів завдань.

Серверна частина містить пакети «Бізнес-логіка», «Звіти та аналітика», «Доступ до даних» і «База даних». Пакет «Бізнес-логіка» координує основні операції системи: роботу із завданнями, користувачами, ролями та статусами. Пакет «Звіти та аналітика» відповідає за формування звітів і розрахунок показників виконання. Пакет «Доступ до даних» виконує запити до бази даних і передає результати іншим модулям. Пакет «База даних» містить структуру таблиць, зв’язки, довідники та збережені записи системи.

Залежності між пакетами відображають напрям взаємодії модулів. Клієнтська частина звертається до серверної логіки через пакет доступу та модулі керування задачами. Серверна бізнес-логіка використовує шар доступу до даних, а звітність і аналітика отримують інформацію з бази через відповідний проміжний пакет. Такий підхід не допускає прямої роботи інтерфейсу з базою даних і забезпечує кращу структурованість програмного забезпечення.

Основні пакети системи можна подати так як представлено у таблиці 2.3.

Таблиця 2.3 – Значення пакетів у системі

| Пакет | Призначення |
|---|---|
| Інтерфейс | Відображення форм, таблиць, дашборду та елементів керування |
| Доступ | Авторизація користувача та перевірка прав |
| Керування задачами | Створення, призначення, перегляд і зміна статусів завдань |
| Бізнес-логіка | Оброблення основних операцій системи |
| Звіти та аналітика | Формування звітів і розрахунок показників |
| Доступ до даних | Виконання запитів до бази даних |
| База даних | Збереження користувачів, ролей, завдань, звітів і показників |

Отже, діаграма пакетів відображає модульну організацію інформаційної системи. Поділ на клієнтську та серверну частини спрощує супровід програмного забезпечення, зменшує залежність між модулями та створює основу для подальшого розширення системи.

## 2.6 Висновки до другого розділу

У другому розділі було виконано проєктування інформаційного та програмного забезпечення інформаційної системи дистанційного керування бізнесом. Основну увагу приділено побудові структури даних, вибору технологічного стеку, визначенню архітектури системи та формуванню її модульної організації.

На основі предметної області розроблено логічну модель даних у вигляді ER-діаграми. У моделі визначено основні сутності системи: ролі, користувачі, завдання, призначення завдань, статуси, звіти, показники та значення показників у звітах. Модель побудована з урахуванням нормалізації даних, що дозволяє зменшити дублювання інформації, забезпечити цілісність зв’язків і підготувати основу для реалізації бази даних.

Також було представлено фізичну модель даних, у якій визначено таблиці, типи полів, первинні й зовнішні ключі, обмеження унікальності та службові поля для фіксації часу створення й оновлення записів. Запропонована структура бази даних забезпечує збереження користувачів, ролей, завдань, статусів виконання,

звітів і аналітичних показників. Для практичної реалізації наведено SQL-фрагменти створення таблиць, заповнення довідників та створення індексів.

Для реалізації програмної системи обґрунтовано вибір технологічного стеку Python, PyQt6 і SQLite. Python використовується для реалізації бізнес-логіки, PyQt6 забезпечує створення графічного інтерфейсу, а SQLite відповідає за локальне збереження даних. Такий стек є доцільним для навчального програмного прототипу, оскільки не потребує складного серверного налаштування, забезпечує швидкий запуск і дозволяє реалізувати повний набір основних функцій системи.

У межах розділу також описано архітектуру системи. Вона передбачає поділ на клієнтський рівень, рівень прикладної логіки та рівень даних. Такий підхід дозволяє відокремити інтерфейс користувача від оброблення бізнес-операцій і роботи з базою даних. Окремо визначено модулі авторизації, керування задачами, користувачами й ролями, звітності, аналітики та доступу до даних.

Діаграма пакетів дала змогу узагальнити програмну структуру системи та показати взаємозв’язки між її основними частинами. Було виділено клієнтську частину з пакетами інтерфейсу, доступу й керування задачами, а також серверну частину з пакетами бізнес-логіки, звітів та аналітики, доступу до даних і бази даних. Такий поділ спрощує супровід системи та створює основу для її подальшого розширення.

Отже, у другому розділі сформовано повну проєктну основу інформаційної системи дистанційного керування бізнесом. Розроблені моделі даних, архітектурні рішення та вибраний технологічний стек забезпечують можливість переходу до програмної реалізації системи, тестування її функцій і подальшого впровадження в середовище малого або середнього бізнесу.

# 3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

## 3.1 Реалізація програмних модулів та інтеграція з базою даних

Програмна реалізація інформаційної системи дистанційного керування бізнесом виконана у вигляді настільного застосунку на Python з використанням бібліотеки PyQt6. Для збереження даних застосовано локальну реляційну базу SQLite, що дозволяє запускати систему без окремого серверного середовища. Основна логіка роботи системи поділена на модулі інтерфейсу, авторизації, роботи з базою даних, керування завданнями, звітності, аналітики та адміністрування.

Запуск системи починається з ініціалізації користувацького інтерфейсу, перевірки наявності бази даних, створення таблиць і завантаження початкових даних. Якщо база відсутня, вона створюється автоматично. Після цього користувач проходить авторизацію та отримує доступ до функцій відповідно до своєї ролі.

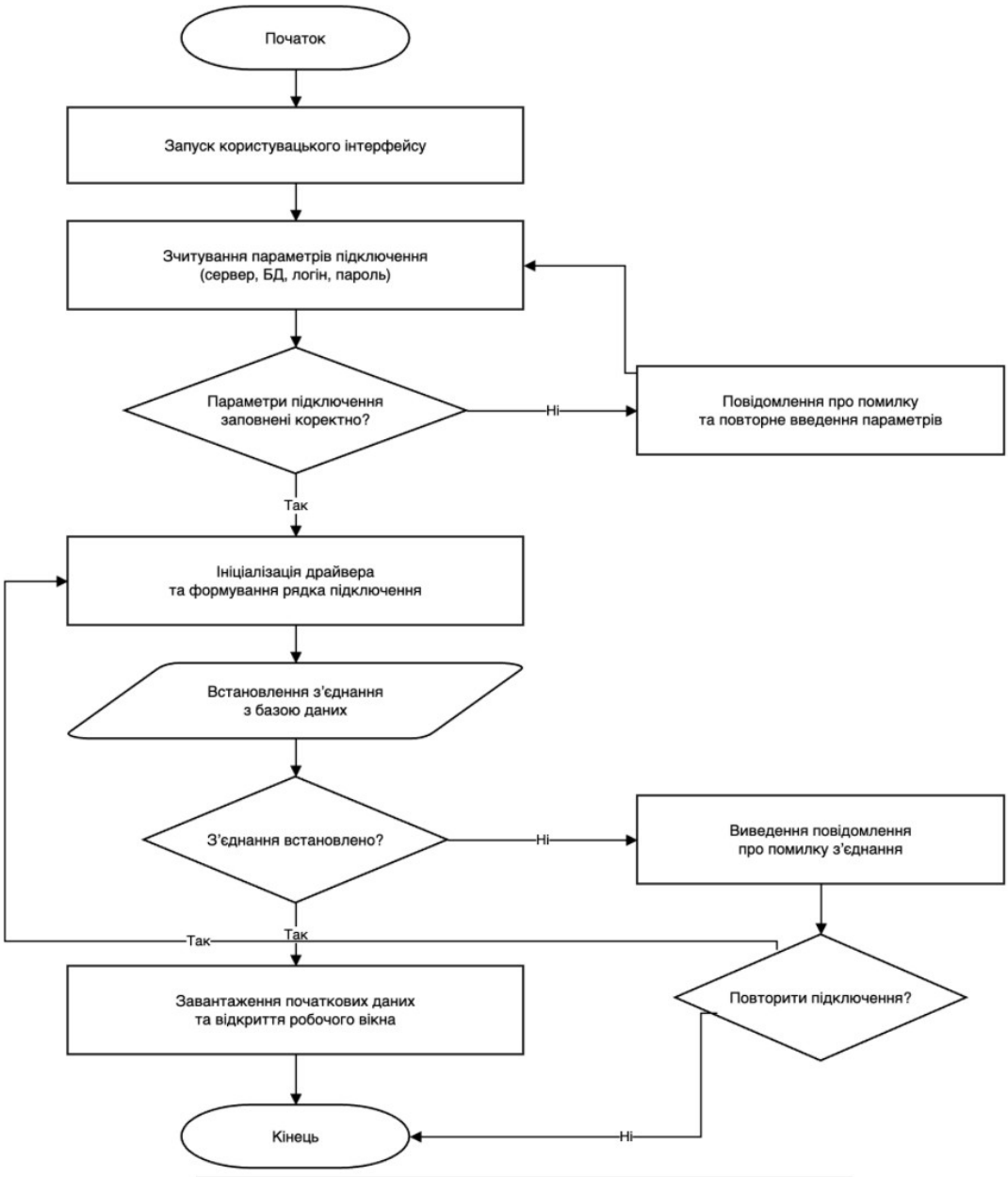


Рисунок 3.1 – Алгоритм запуску системи та підключення до бази даних

Структура програмних модулів наведена в таблиці 3.1.

Таблиця 3.1 – Основні програмні модулі системи

| Модуль | Файл | Призначення |
|---|---|---|
| Головний модуль | main.py | Запуск програми, формування інтерфейсу, обробка дій користувача |
| Модуль бази даних | database.py | Створення таблиць, SQL-запити, початкове заповнення даних |
| Модуль авторизації | auth.py | Хешування паролів і перевірка облікових даних |
| Модуль задач | main.py, database.py | Створення, призначення, перегляд і оновлення статусів завдань |

Продовження таблиці 3.1

| | | |
|---|---|---|
| Модуль звітності | database.py | Формування статистики за завданнями та показниками виконання |
| Модуль адміністрування | main.py, database.py | Керування користувачами, ролями, довідниками й журналом дій |

Інтеграція з базою даних реалізована через окремий клас роботи з SQLite. Такий підхід дозволяє не змішувати SQL-запити з елементами інтерфейсу та спрощує супровід програми. У базі даних зберігаються користувачі, ролі, завдання, призначення, статуси, звіти, показники, повідомлення та журнал дій.

Фрагмент створення з'єднання з базою даних має такий вигляд:

```
import sqlite3

class Database:
    def __init__(self, db_path="remote_business.db"):
        self.db_path = db_path
        self.connection = sqlite3.connect(self.db_path)
        self.connection.row_factory = sqlite3.Row

    def execute(self, query, params=()):
        cursor = self.connection.cursor()
        cursor.execute(query, params)
        self.connection.commit()
        return cursor
```

Після встановлення з'єднання система створює необхідні таблиці, якщо вони ще не існують. Основними таблицями є roles, users, tasks, task_statuses, task_assignments, reports, indicators, notifications та activity_logs. Для підтримки цілісності використовуються первинні ключі, зовнішні ключі та унікальні обмеження.

Авторизація користувача реалізована через перевірку логіна, пароля та активності облікового запису. Паролі не зберігаються у відкритому вигляді, а порівнюються після хешування. Після успішного входу система визначає роль користувача та відкриває відповідний набір вкладок.

```
def login_user(self, email, password_hash):
    query = """
        SELECT users.*, roles.role_name
        FROM users
        JOIN roles ON roles.role_id = users.role_id
        WHERE users.email = ? AND users.password_hash = ? AND users.is_active =
1
    """
    return self.execute(query, (email, password_hash)).fetchone()
```

Рольова модель доступу є однією з основних частин реалізації. Власник бізнесу отримує доступ до дашборду, аналітики, контролю виконання та звітів. Менеджер працює зі створенням і призначенням завдань, контролем строків і формуванням звітів. Співробітник переглядає власні завдання та змінює їх статус. Адміністратор керує користувачами, ролями, довідниками та журналом дій.

Модуль керування завданнями забезпечує створення нового завдання, вибір виконавця, встановлення пріоритету, дедлайну та поточного статусу. Після збереження завдання система створює запис у таблиці tasks, а також запис про призначення у таблиці task_assignments.

```
def create_task(self, created_by, title, description, priority, due_at):
    query = """
        INSERT INTO tasks(created_by, task_title, task_description, priority,
due_at)
        VALUES (?, ?, ?, ?, ?)
    """
    cursor = self.execute(query, (created_by, title, description, priority,
due_at))
    return cursor.lastrowid
```

Оновлення статусу виконання виконується співробітником або менеджером залежно від прав доступу. Система фіксує новий статус, коментар до виконання та дату завершення, якщо завдання переведено у стан “Виконано”.

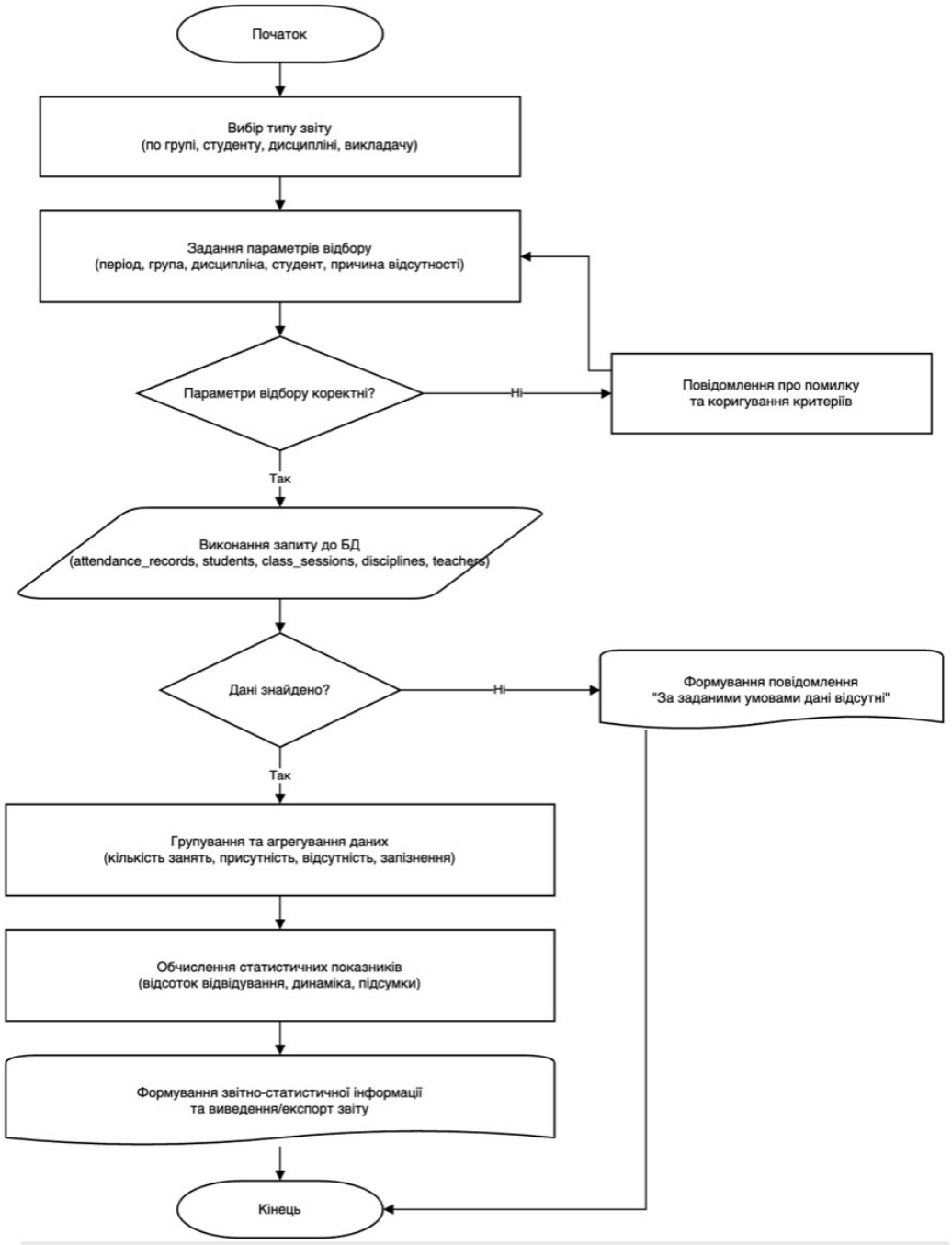


Рисунок 3.2 – Алгоритм оновлення статусу виконання завдання

Для формування звітності система використовує дані з таблиць tasks, task_assignments і task_statuses. Обчислюються кількість усіх завдань, кількість виконаних, активних і прострочених завдань, а також рівень виконання за вибраний період. Результати можуть відображатися в інтерфейсі або експортуватися у CSV-файл.

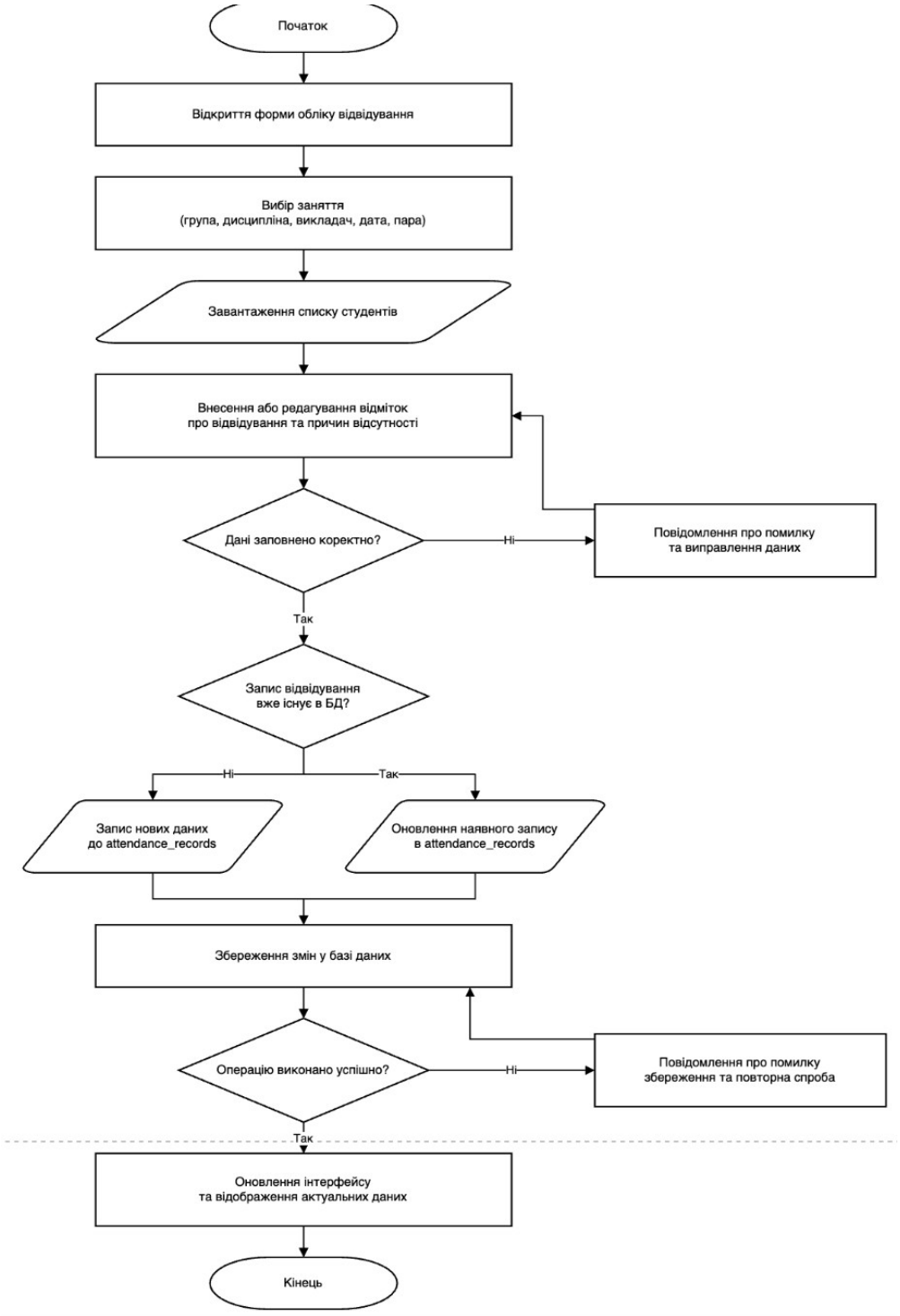


Рисунок 3.3 – Алгоритм формування аналітичного звіту

Приклад SQL-запиту для отримання показників виконання:

```
SELECT
    COUNT(*) AS total_tasks,
    SUM(CASE WHEN ts.status_name = 'Виконано' THEN 1 ELSE 0 END) AS
completed_tasks,
    SUM(CASE WHEN t.due_at < CURRENT_TIMESTAMP
             AND ts.status_name <> 'Виконано' THEN 1 ELSE 0 END) AS
overdue_tasks
FROM task_assignments ta
```

```
JOIN tasks t ON t.task_id = ta.task_id
JOIN task_statuses ts ON ts.status_id = ta.status_id;
```

Інтерфейс системи реалізовано засобами PyQt6 з використанням вкладок, таблиць, форм введення та інформаційних карток. Для покращення вигляду застосовано QSS-стилі, які задають оформлення кнопок, таблиць, панелей, заголовків і полів введення. Це дозволяє зробити інтерфейс більш читабельним і зручним для користувачів з різними ролями.

У процесі реалізації передбачено базові механізми контролю помилок. Система перевіряє заповнення обов’язкових полів, коректність введених даних, наявність вибраного виконавця, статусу та дедлайну. У разі помилки користувач отримує повідомлення, а некоректна операція не записується до бази даних.

Отже, у межах реалізації програмних модулів було створено функціональний прототип інформаційної системи дистанційного керування бізнесом. Програма забезпечує авторизацію користувачів, розмежування доступу за ролями, керування завданнями, оновлення статусів, формування звітів, перегляд аналітичних показників і роботу з локальною базою даних. Реалізована структура є достатньою для подальшого тестування системи та оцінювання її працездатності.

### 3.2 Проведення тестування системи

Тестування інформаційної системи дистанційного керування бізнесом виконувалося з метою перевірки працездатності основних програмних модулів, коректності взаємодії з базою даних, правильності рольового доступу та зручності користувацького інтерфейсу. Основну увагу приділено перевірці авторизації, роботи дашборду, керування завданнями, формування звітів, перегляду персоналу, журналу дій і профілю користувача.

Першим етапом було перевірено запуск програми та роботу форми входу. На екрані авторизації користувач може вибрати демонстраційний обліковий запис, ввести email і пароль, після чого система виконує перевірку даних у

локальній SQLite-базі. У разі успішного входу відкривається робоче вікно відповідно до ролі користувача.

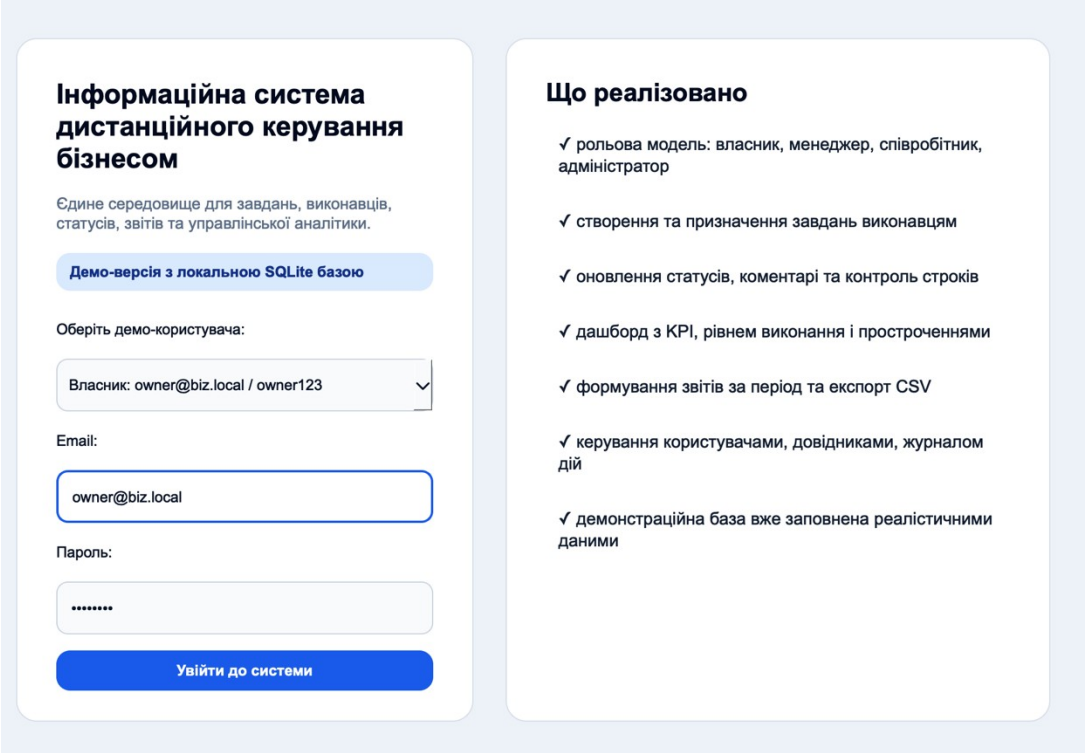


Рисунок 3.4 – Вікно авторизації користувача в інформаційній системі

Після авторизації власника бізнесу було перевірено роботу головної панелі огляду системи. На дашборді відображаються ключові показники: загальна кількість завдань, кількість виконаних, активних і прострочених завдань. Також виводиться рівень виконання у відсотках, таблиця останніх завдань і блок повідомлень. Це підтверджує коректність отримання агрегованих даних із бази.

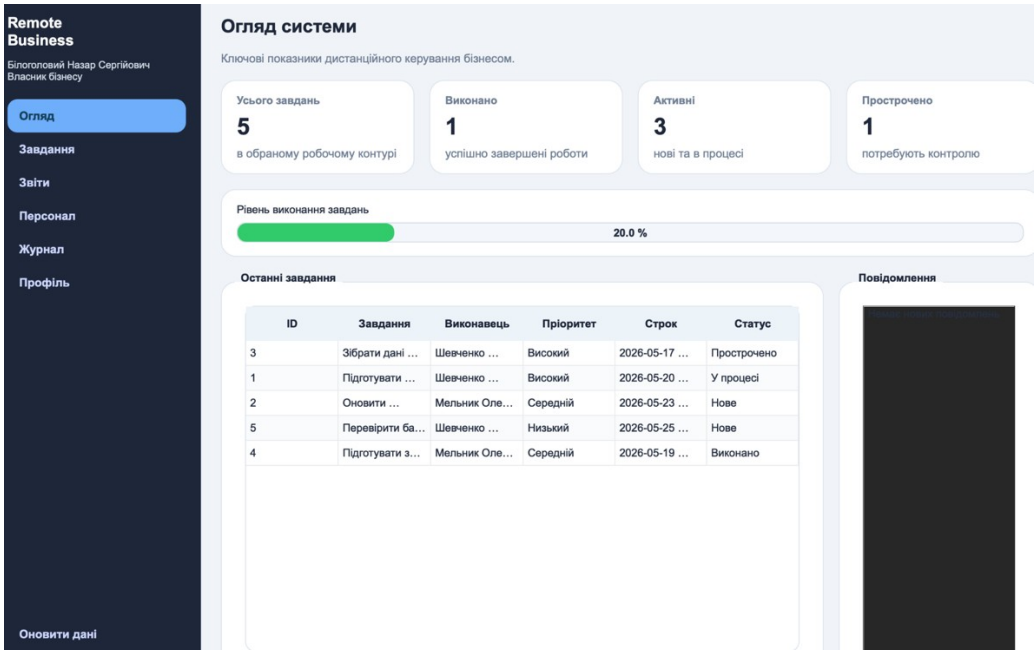


Рисунок 3.5 – Головна панель огляду системи з ключовими показниками

Наступним етапом перевірено модуль керування завданнями. У цьому модулі користувач із відповідними правами може створити нове завдання, вказати назву, строк, пріоритет, виконавця та опис роботи. Після збереження завдання запис додається до бази даних і відображається у таблиці. Також перевірено зміну статусу завдання та експорт даних у CSV.

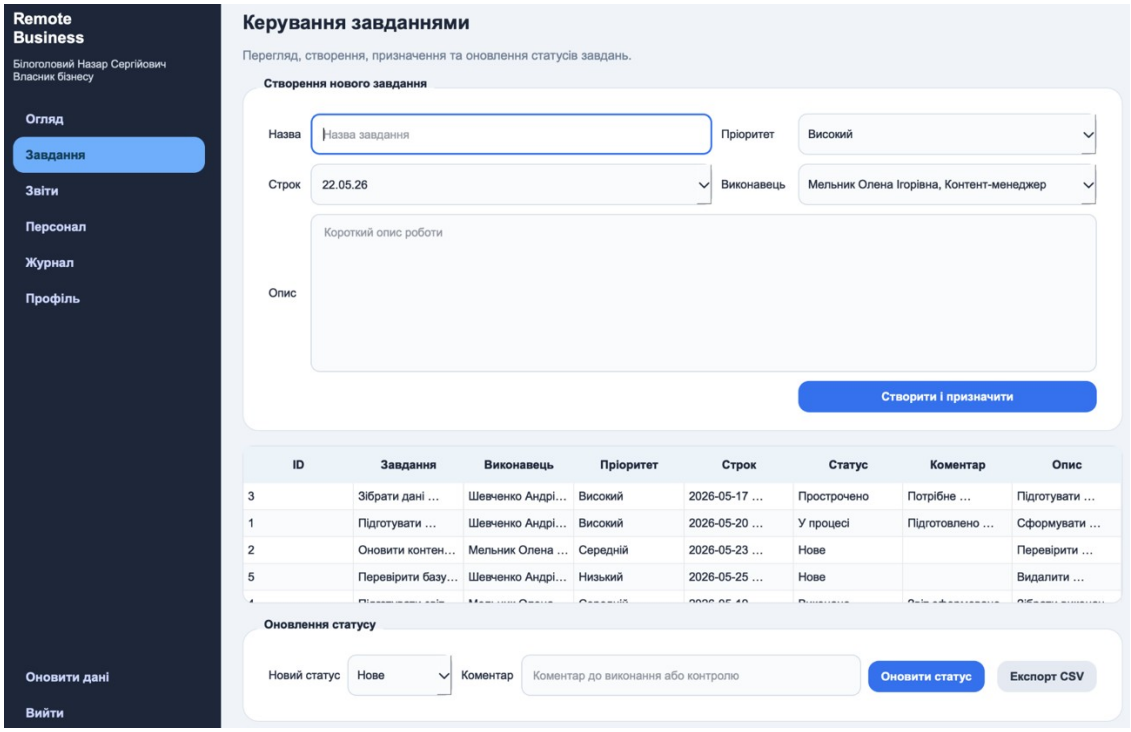


Рисунок 3.6 – Модуль створення, призначення та оновлення статусу завдань

Окремо протестовано модуль звітів та аналітики. Користувач задає назву звіту, тип, початкову та кінцеву дату періоду. Після натискання кнопки формування система обчислює кількість завдань, кількість виконаних і прострочених робіт, а також відсоток виконання. Отримані дані зберігаються в базі та відображаються у таблиці звітів.

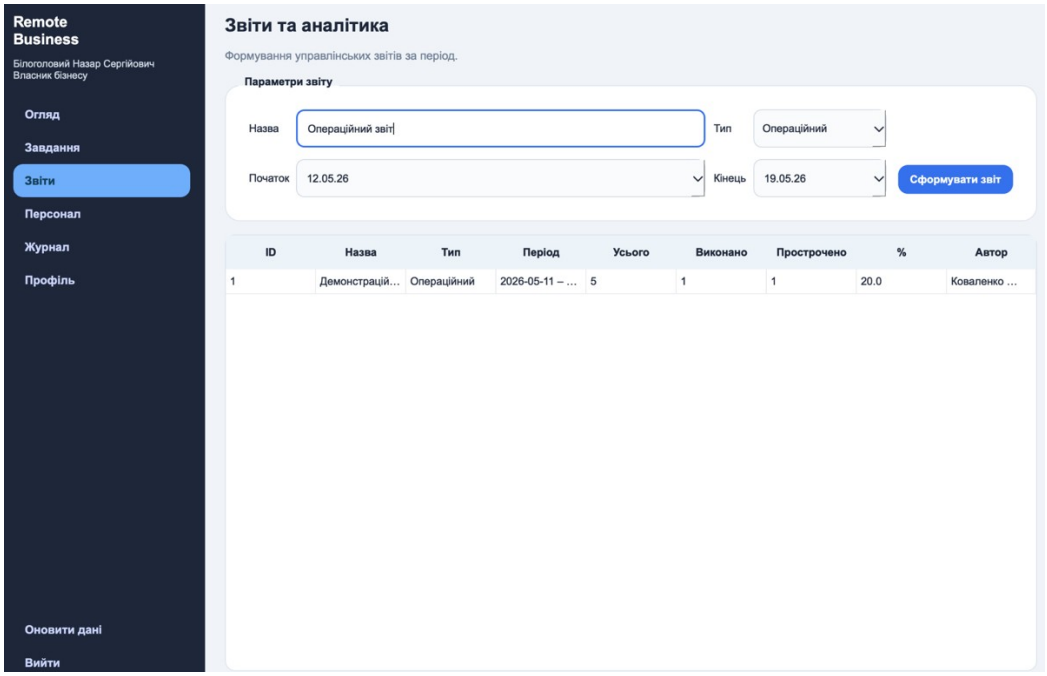


Рисунок 3.7 – Модуль формування управлінських звітів

Перевірка вкладки персоналу показала, що система коректно відображає користувачів, їх ролі, посади, email, телефони, активність і результативність виконання завдань. Це дає змогу власнику бізнесу або адміністратору переглядати поточний склад персоналу та контролювати участь співробітників у виконанні робіт.

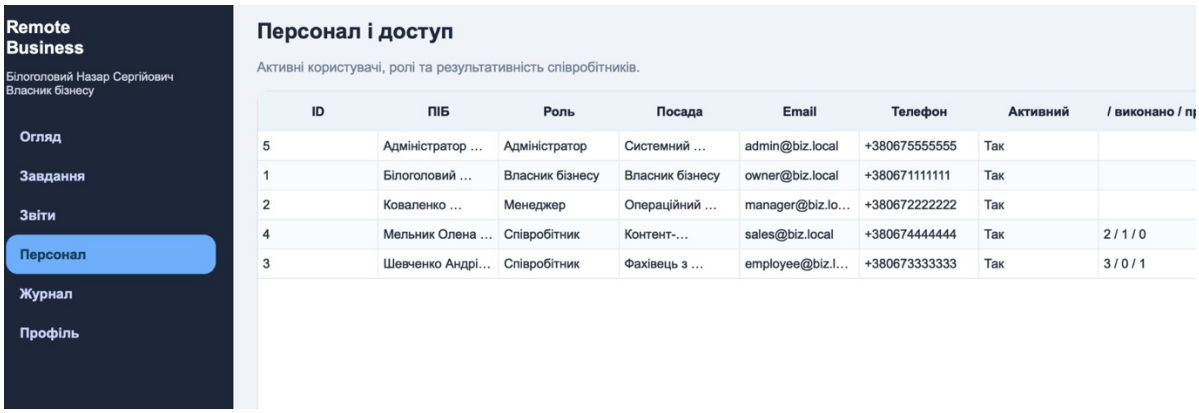


Рисунок 3.8 – Вікно перегляду персоналу та ролей користувачів

Також було протестовано вкладку журналу дій і довідників. У системі відображаються ролі користувачів, статуси завдань і журнал операцій. Журнал фіксує важливі події, зокрема вхід користувача, створення звіту та ініціалізацію бази даних. Це підтверджує роботу механізму контролю дій користувачів.

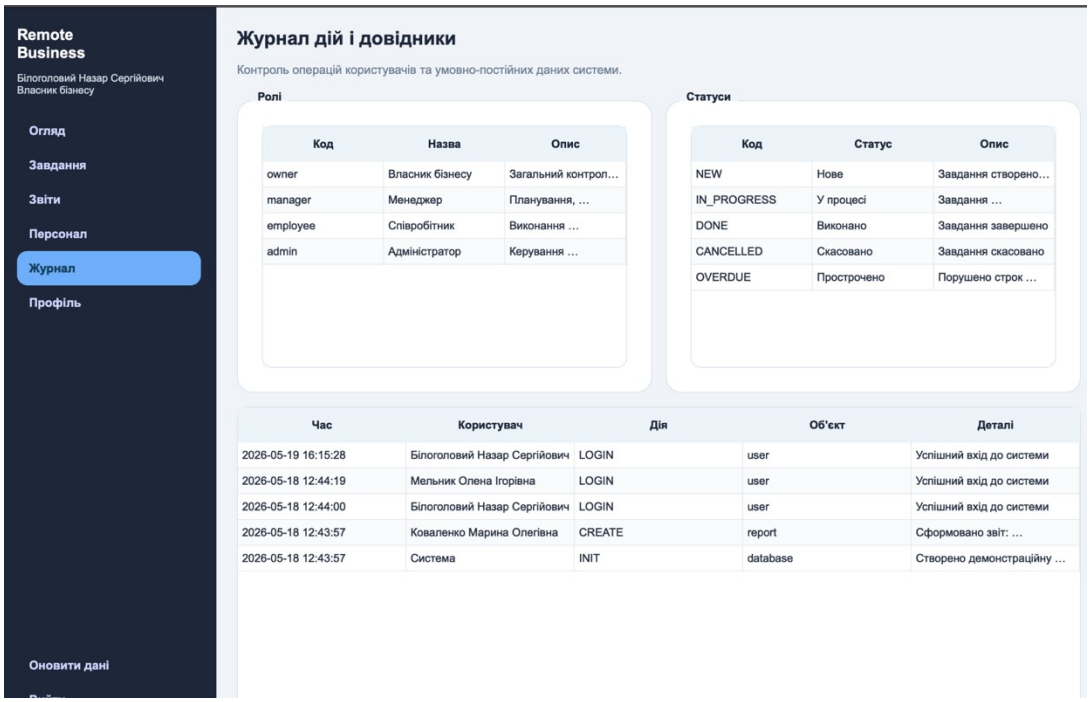


Рисунок 3.9 – Вікно журналу дій і довідників системи

Останнім етапом перевірено відображення профілю користувача. У профілі виводяться ПІБ, email, роль, посада, телефон і дата створення облікового запису. Дані відповідають інформації, що зберігається в базі, тому модуль профілю працює коректно.

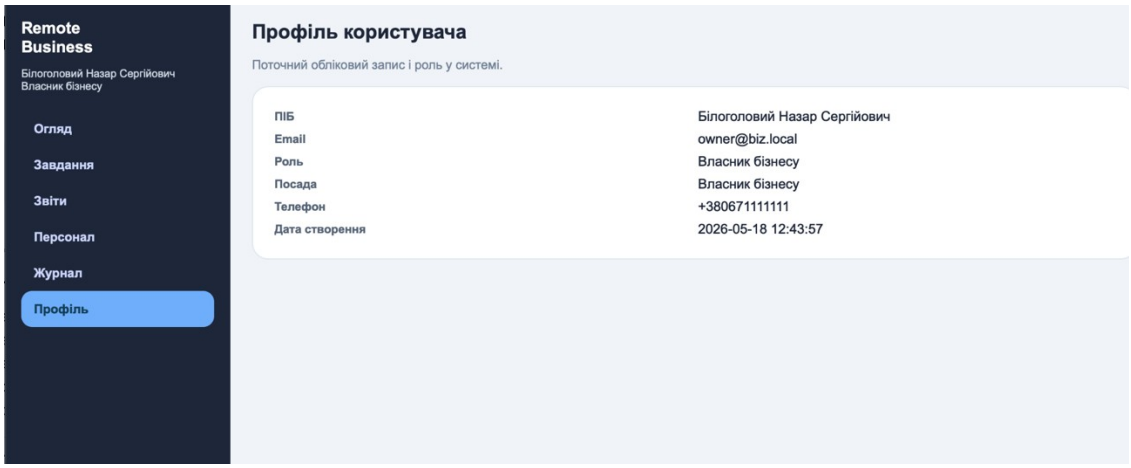


Рисунок 3.10 – Вікно профілю поточного користувача

План і результати основних тестів наведено в таблиці 3.2.

Таблиця 3.2 – Результати тестування функціональних модулів системи

| № | Модуль | Перевірена дія | Очікуваний результат | Фактичний результат | Статус |
|---|---|---|---|---|---|
| 1 | Авторизація | Вхід за email і паролем | Відкриття системи відповідно до ролі | Користувач успішно входить у систему | Виконано |
| 2 | Рольовий доступ | Вхід під різними ролями | Відображення дозволених функцій | Для ролей відкриваються відповідні вкладки | Виконано |
| 3 | Дашборд | Перегляд KPI | Виведення кількості завдань і рівня виконання | Показники відображаються коректно | Виконано |
| 4 | Завдання | Створення нового завдання | Запис завдання в БД і поява в таблиці | Завдання створюється та відображається | Виконано |
| 5 | Призначення | Вибір виконавця для завдання | Завдання закріплюється за співробітником | Виконавець зберігається правильно | Виконано |
| 6 | Статус | Оновлення статусу виконання | Зміна статусу в таблиці та БД | Статус оновлюється після підтвердження | Виконано |
| 7 | Звіти | Формування звіту за період | Створення запису зі статистикою | Звіт формується і додається в таблицю | Виконано |
| 8 | Персонал | Перегляд користувачів | Відображення ролей, контактів і активності | Дані персоналу завантажуються з БД | Виконано |
| 9 | Журнал | Фіксація дій користувачів | Запис подій у журнал | Події LOGIN, CREATE та INIT відображаються | Виконано |
| 10 | Профіль | Перегляд даних користувача | Виведення облікової інформації | Дані профілю відображаються коректно | Виконано |
| 11 | Експорт | Експорт завдань у CSV | Створення файлу з даними таблиці | Експорт виконується без помилок | Виконано |
| 12 | Перевірка помилок | Порожні або некоректні поля | Повідомлення про помилку | Некоректні операції не зберігаються | Виконано |

Під час тестування було підтверджено, що система коректно працює з локальною базою даних. Дані про користувачів, ролі, завдання, статуси, звіти та журнал дій зберігаються у відповідних таблицях і відображаються в інтерфейсі після оновлення. Це свідчить про правильну інтеграцію програмних модулів із SQLite.

Окремо перевірено зручність інтерфейсу. Усі основні функції згруповані у боковому меню: огляд, завдання, звіти, персонал, журнал і профіль. Така структура дозволяє швидко переходити між модулями. Контрастне оформлення, великі заголовки, таблиці та виділення активної вкладки покращують сприйняття інформації та зменшують ризик помилок під час роботи.

За результатами тестування встановлено, що розроблена інформаційна система виконує основні функції, визначені у вимогах: авторизацію користувачів, розмежування доступу за ролями, керування завданнями, контроль статусів, формування звітів, перегляд персоналу, ведення журналу дій і роботу з профілем користувача. Отримані результати підтверджують працездатність системи та її придатність для використання як програмного прототипу інформаційної системи дистанційного керування бізнесом.

### 3.3 Розгортання системи, склад інсталяційного пакету

Розгортання інформаційної системи дистанційного керування бізнесом виконано у вигляді локального desktop-застосунку. Такий варіант є доцільним для програмного прототипу, оскільки не потребує окремого веб-сервера, хостингу або складного адміністрування. Усі основні компоненти системи запускаються на робочому комп’ютері користувача, а дані зберігаються у локальній базі SQLite.

На рисунку 3.11 подано діаграму розгортання інформаційної системи дистанційного керування бізнесом.

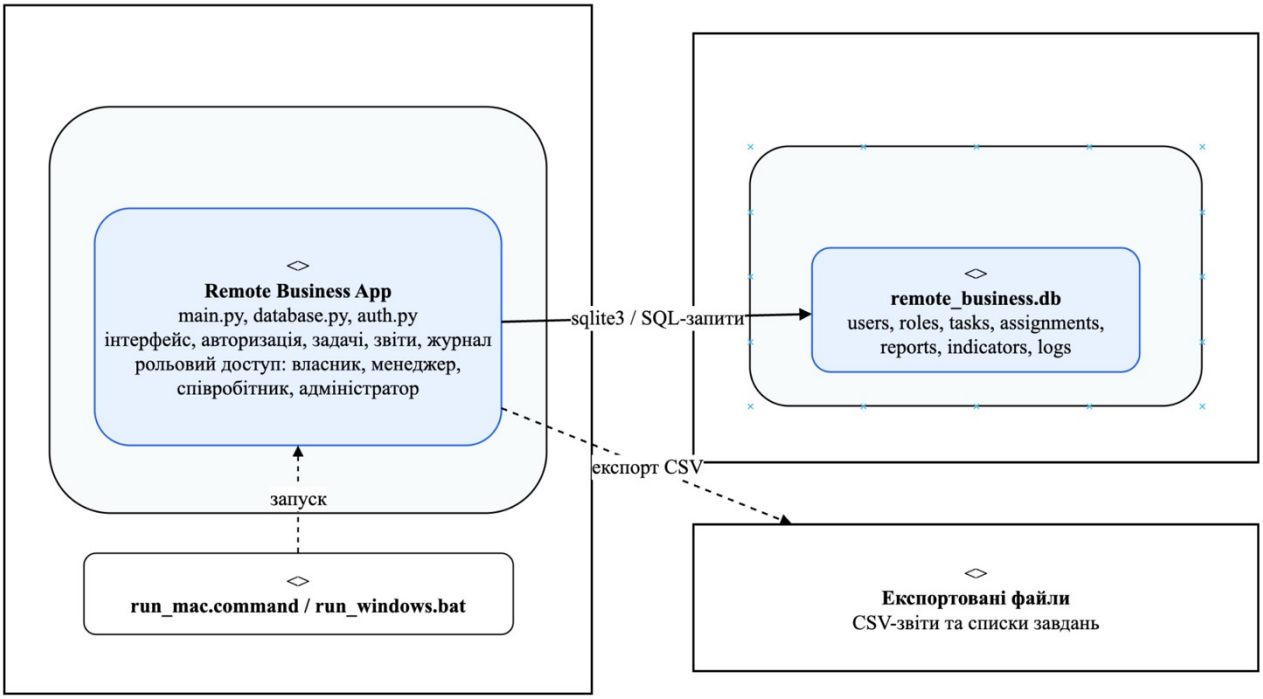


Рисунок 3.11 – Діаграма розгортання інформаційної системи дистанційного керування бізнесом

Діаграма показує, що основним виконуваним компонентом є застосунок Remote Business App, який містить файли main.py, database.py та auth.py. У межах цього компонента реалізовано графічний інтерфейс, авторизацію користувачів, керування завданнями, формування звітів, журнал дій та рольове розмежування доступу для власника бізнесу, менеджера, співробітника й адміністратора.

Застосунок взаємодіє з локальною базою даних remote_business.db через вбудований механізм sqlite3. У базі зберігаються таблиці користувачів, ролей, завдань, призначень, статусів, звітів, показників і журналу дій. Передавання даних між програмою та базою відбувається через SQL-запити, які виконуються безпосередньо з модуля роботи з базою даних.

Окремо передбачено експорт результатів у CSV-файли. Це дозволяє зберігати списки завдань або звітні дані у форматі, який можна відкрити в табличних редакторах, наприклад Microsoft Excel або LibreOffice Calc. Такий механізм є корисним для подальшого аналізу результатів роботи системи або передачі звітів керівнику.

Склад інсталяційного пакету наведено в таблиці 3.3.

Таблиця 3.3 – Склад інсталяційного пакету системи

| Компонент | Призначення |
|---|---|
| main.py | Головний файл запуску програми та реалізації інтерфейсу |
| database.py | Модуль створення бази даних, таблиць і виконання SQL-запитів |
| auth.py | Модуль авторизації та перевірки паролів |
| remote_business.db | Локальна база даних SQLite |
| requirements.txt | Перелік необхідних Python-бібліотек |
| run_windows.bat | Файл швидкого запуску програми в середовищі Windows |
| run_mac.command | Файл швидкого запуску програми в середовищі macOS |
| README.md | Коротка інструкція з запуску та використання системи |
| exports/ | Каталог для збереження експортованих CSV-звітів |

Для запуску системи на комп’ютері користувача необхідно встановити Python, встановити залежності з файлу requirements.txt і запустити головний файл програми. У разі першого запуску база даних створюється автоматично, після чого система заповнюється демонстраційними користувачами, ролями, статусами та тестовими завданнями.

Послідовність розгортання системи включає такі етапи:

- копіювання папки проєкту на робочий комп’ютер;
- встановлення Python та необхідних бібліотек;
- запуск файлу run_windows.bat або run_mac.command;
- автоматичне створення локальної бази даних;
- завантаження демонстраційних даних;
- авторизація користувача та перевірка роботи системи.

Мінімальні вимоги до середовища запуску є невисокими. Для роботи достатньо комп’ютера з операційною системою Windows або macOS, встановленим Python 3 та бібліотекою PyQt6. Оскільки база даних SQLite зберігається локально, додатковий сервер бази даних не потрібний.

Перевагою такого варіанта розгортання є простота встановлення та автономність роботи. Користувач може запустити систему без налаштування

мережевого сервера, облікових записів хостингу або зовнішньої СУБД. Це спрощує тестування програмного продукту та демонстрацію його основних можливостей.

Водночас така схема розгортання має певні обмеження. Локальна база даних зручна для прототипу та індивідуального використання, але для багатокористувацької експлуатації в реальному бізнес-середовищі доцільно перенести систему на клієнт-серверну архітектуру з окремим сервером застосунків і централізованою базою даних. У такому разі SQLite може бути замінено на PostgreSQL або MySQL, а desktop-інтерфейс може бути доповнений веб-версією.

Отже, розгортання системи реалізовано у компактному локальному форматі, який відповідає вимогам програмного прототипу. Інсталяційний пакет містить усі необхідні файли для запуску, створення бази даних, авторизації користувачів, роботи із завданнями, звітами та журналом дій. Обрана схема забезпечує простий запуск, автономність роботи й можливість подальшого розширення системи.

### 3.4 Висновки до третього розділу

У третьому розділі було розглянуто процес реалізації та тестування інформаційної системи дистанційного керування бізнесом. Основну увагу приділено практичній побудові програмних модулів, інтеграції з базою даних, перевірці працездатності інтерфейсу та оцінюванню коректності виконання основних функцій системи.

Програмну реалізацію виконано у вигляді desktop-застосунку на основі Python, PyQt6 та SQLite. Такий підхід дозволив створити автономну систему, яка не потребує складного серверного розгортання та може запускатися на робочому комп’ютері користувача. У системі реалізовано авторизацію, рольове розмежування доступу, керування завданнями, формування звітів, перегляд персоналу, журнал дій, профіль користувача та експорт даних у CSV.

Під час реалізації було забезпечено інтеграцію програмних модулів із локальною базою даних. У базі зберігаються користувачі, ролі, завдання, призначення, статуси, звіти, показники та службові записи журналу. Програма автоматично створює базу даних під час першого запуску та заповнює її демонстраційною інформацією, що спрощує перевірку функціональності системи.

У межах тестування перевірено основні сценарії роботи програми: вхід користувача до системи, відкриття функцій відповідно до ролі, перегляд дашборду, створення та призначення завдань, оновлення статусів, формування звітів, перегляд персоналу, ведення журналу дій і відображення профілю користувача. Результати тестування підтвердили, що система коректно обробляє дані, зберігає зміни в базі та відображає актуальну інформацію в інтерфейсі.

Також було розглянуто розгортання системи та склад інсталяційного пакету. До складу пакету входять основні програмні файли, модуль бази даних, файл залежностей, скрипти запуску для Windows і macOS, локальна база SQLite та каталог для експортованих CSV-файлів. Запропонована схема розгортання є простою, зрозумілою та достатньою для демонстраційного прототипу.

Отже, у третьому розділі підтверджено працездатність розробленої інформаційної системи дистанційного керування бізнесом. Реалізований програмний продукт забезпечує виконання ключових функцій, визначених у попередніх розділах, має зручний інтерфейс, підтримує роботу з базою даних і може бути використаний як основа для подальшого розвитку, зокрема переходу до клієнт-серверної або веб-архітектури.

# ВИСНОВКИ

У кваліфікаційній роботі було виконано проєктування та розроблення інформаційної системи дистанційного керування бізнесом. Актуальність роботи зумовлена потребою підприємств у зручних програмних засобах для організації роботи персоналу, контролю виконання завдань, формування звітності та перегляду ключових показників діяльності в дистанційному режимі.

У першому розділі було досліджено предметну область дистанційного керування бізнесом. Визначено, що основними учасниками процесу є власник бізнесу, менеджер, співробітник та адміністратор. Проаналізовано типові проблеми організації управління за допомогою фрагментарних інструментів, зокрема електронних таблиць, месенджерів і окремих сервісів для задач. Встановлено, що такі підходи не завжди забезпечують цілісний контроль виконання робіт, узгодженість даних і швидке формування аналітичної інформації. Також було розглянуто існуючі програмні рішення, серед яких Trello, Asana, ClickUp, monday.com та Bitrix24. Проведений аналіз показав доцільність створення власної системи, орієнтованої на просту рольову модель, облік завдань, контроль статусів і формування управлінських звітів.

У межах моделювання предметної області було побудовано DFD-діаграму нульового рівня, діаграму прецедентів, діаграму послідовності та діаграму активності. Ці моделі дозволили формалізувати основні сценарії роботи системи: авторизацію користувачів, перегляд показників бізнесу, створення й призначення завдань, оновлення статусів, контроль виконання, формування звітів та управління персоналом. На основі виконаного аналізу було визначено функціональні, нефункціональні, технічні та безпекові вимоги до програмної системи.

У другому розділі було виконано проєктування інформаційного та програмного забезпечення. Розроблено логічну модель даних у вигляді ER-діаграми, у якій визначено сутності ролей, користувачів, завдань, призначень,

статусів, звітів і показників. Модель приведено до третьої нормальної форми, що дозволяє зменшити дублювання даних і забезпечити цілісність зв’язків між об’єктами. На її основі побудовано фізичну модель бази даних із визначенням таблиць, ключів, обмежень, індексів і SQL-запитів для створення структури та заповнення довідників.

Для реалізації системи було обґрунтовано вибір технологічного стеку Python, PyQt6 та SQLite. Python використано для реалізації бізнес-логіки, PyQt6 для створення графічного інтерфейсу, а SQLite для локального збереження даних. Такий стек забезпечив простий запуск системи, автономність роботи, зручне тестування та достатню функціональність для програмного прототипу. Також було описано архітектуру системи, її основні рівні та модулі: інтерфейс, авторизація, керування задачами, користувачі й ролі, звіти, аналітика, доступ до даних і база даних. Діаграма пакетів відобразила модульну організацію програмного забезпечення.

У третьому розділі було реалізовано програмний прототип інформаційної системи дистанційного керування бізнесом. Розроблена програма підтримує авторизацію користувачів, рольове розмежування доступу, дашборд із ключовими показниками, створення та призначення завдань, оновлення статусів, формування звітів, перегляд персоналу, журнал дій, профіль користувача та експорт даних у CSV. База даних створюється автоматично під час запуску та містить демонстраційні дані для перевірки роботи системи.

Проведене тестування підтвердило працездатність основних функціональних модулів. Перевірено вхід користувачів за ролями, відображення показників на дашборді, створення завдань, призначення виконавців, зміну статусів, формування звітів, перегляд персоналу, ведення журналу дій і відображення профілю користувача. Результати тестування показали, що система коректно взаємодіє з базою даних, зберігає зміни та відображає актуальну інформацію в інтерфейсі.

Також було розглянуто процес розгортання системи та склад інсталяційного пакету. До складу програмного продукту входять основні файли

застосунку, модуль бази даних, модуль авторизації, файл залежностей, скрипти запуску для Windows і macOS, локальна база SQLite та каталог для експортованих файлів. Обрана схема розгортання є простою та зручною для демонстраційного використання, оскільки не потребує окремого сервера або складного налаштування середовища.

Отже, мету кваліфікаційної роботи досягнуто. Було спроєктовано та реалізовано інформаційну систему дистанційного керування бізнесом, яка забезпечує автоматизацію основних управлінських процесів: облік користувачів, керування завданнями, контроль виконання, формування звітності та перегляд аналітичних показників. Розроблений програмний продукт може бути використаний як основа для подальшого розвитку, зокрема переходу до повноцінної клієнт-серверної або веб-архітектури, інтеграції з хмарною базою даних, додавання системи повідомлень і розширення аналітичного модуля.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 816 p. URL: https://www.pearson.com/en-us/subject-catalog/p/software-engineering/P200000003258/9780137503148
2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. McGraw-Hill Education, 2019. 672 p. URL: https://www.mheducation.com/highered/product/software-engineering-a-practitioners-approach-pressman.html
3. ISO/IEC/IEEE 29148:2018. Systems and software engineering. Life cycle processes. Requirements engineering. ISO, 2018. URL: https://www.iso.org/standard/72089.html
4. IEEE SA. ISO/IEC/IEEE 29148-2018. Systems and software engineering. Life cycle processes. Requirements engineering. URL: https://standards.ieee.org/standard/29148-2018.html
5. Object Management Group. Unified Modeling Language, Version 2.5.1. OMG, 2017. URL: https://www.omg.org/spec/UML/2.5.1/About-UML
6. Object Management Group. Unified Modeling Language Specification. URL: https://www.omg.org/spec/UML/
7. Python Software Foundation. Python 3 Documentation. URL: https://docs.python.org/3/
8. Python Software Foundation. Python Programming Language. Official Website. URL: https://www.python.org/
9. Riverbank Computing. PyQt6 Documentation. URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/
10. Riverbank Computing. PyQt. Introduction and Components. URL: https://riverbankcomputing.com/software/pyqt/intro
11. Python Package Index. PyQt6. URL: https://pypi.org/project/PyQt6/
12. Qt Group. Qt for Python Documentation. URL: https://doc.qt.io/qtforpython-6/

13. Qt Group. Qt Style Sheets Reference. URL: https://doc.qt.io/qt-6/stylesheet-reference.html
14. Qt Group. Qt Style Sheets. URL: https://doc.qt.io/qt-6/stylesheet.html
15. SQLite Consortium. SQLite Documentation. URL: https://www.sqlite.org/docs.html
16. SQLite Consortium. SQLite Home Page. URL: https://sqlite.org/
17. SQLite Consortium. SQL As Understood By SQLite. URL: https://www.sqlite.org/lang.html
18. SQLite Consortium. SQLite Foreign Key Support. URL: https://www.sqlite.org/foreignkeys.html
19. DB Browser for SQLite. DB Browser for SQLite Documentation. URL: https://sqlitebrowser.org/
20. OWASP Foundation. Application Security Verification Standard. URL: https://owasp.org/www-project-application-security-verification-standard/
21. OWASP Foundation. OWASP ASVS GitHub Repository. URL: https://github.com/OWASP/ASVS
22. National Institute of Standards and Technology. Cybersecurity Framework. URL: https://www.nist.gov/cyberframework
23. National Institute of Standards and Technology. Secure Software Development Framework, NIST SP 800-218. URL: https://csrc.nist.gov/publications/detail/sp/800-218/final
24. Atlassian. Trello. Official Website. URL: https://trello.com/
25. Atlassian Support. Add a due date or start date to a card. Trello Documentation. URL: https://support.atlassian.com/trello/docs/adding-dates-to-cards/
26. Asana. Manage your team's work, projects, and tasks online. URL: https://asana.com/
27. Asana. Project Management Features. URL: https://asana.com/features/project-management
28. Asana Help Center. Using Asana for project management. URL: https://help.asana.com/s/article/managing-projects

29. ClickUp. ClickUp. The everything app for work. URL: https://clickup.com/
30. ClickUp Help Center. Use Dashboards for goals and OKRs. URL: https://help.clickup.com/hc/en-us/articles/30807421482391-Use-Dashboards-for-goals-and-OKRs
31. ClickUp Help Center. Create views for project management. URL: https://help.clickup.com/hc/en-us/articles/9764027803415-Create-views-for-project-management
32. monday.com. Work Management Platform. URL: https://monday.com/
33. monday.com. Dashboards and Reporting. URL: https://monday.com/w/dashboards
34. monday.com Support. The Dashboards. URL: https://support.monday.com/hc/en-us/articles/360002187819-The-Dashboards
35. Bitrix24. Free online workspace for business. URL: https://www.bitrix24.com/
36. Bitrix24. Tasks and Projects. URL: https://www.bitrix24.com/tools/tasks_and_projects/
37. Microsoft. Visual Studio Code Documentation. URL: https://code.visualstudio.com/docs
38. Git Documentation. Git Reference Manual. URL: https://git-scm.com/docs
39. Python Software Foundation. sqlite3. DB-API 2.0 interface for SQLite databases. URL: https://docs.python.org/3/library/sqlite3.html
40. Python Software Foundation. csv. CSV File Reading and Writing. URL: https://docs.python.org/3/library/csv.html

# ДОДАТОК А

SQL-код створення бази даних інформаційної системи дистанційного керування бізнесом

```
PRAGMA foreign_keys = ON;

CREATE TABLE IF NOT EXISTS roles (
    role_id INTEGER PRIMARY KEY AUTOINCREMENT,
    role_name TEXT NOT NULL UNIQUE,
    description TEXT
);

CREATE TABLE IF NOT EXISTS users (
    user_id INTEGER PRIMARY KEY AUTOINCREMENT,
    role_id INTEGER NOT NULL,
    full_name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    password_hash TEXT NOT NULL,
    phone TEXT,
    is_active INTEGER NOT NULL DEFAULT 1,
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (role_id) REFERENCES roles(role_id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE IF NOT EXISTS task_statuses (
    status_id INTEGER PRIMARY KEY AUTOINCREMENT,
    status_code TEXT NOT NULL UNIQUE,
    status_name TEXT NOT NULL,
    description TEXT
);

CREATE TABLE IF NOT EXISTS tasks (
    task_id INTEGER PRIMARY KEY AUTOINCREMENT,
    created_by INTEGER NOT NULL,
    title TEXT NOT NULL,
    description TEXT,
    priority TEXT NOT NULL DEFAULT 'Середній',
    due_at TEXT NOT NULL,
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TEXT,
    FOREIGN KEY (created_by) REFERENCES users(user_id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE IF NOT EXISTS task_assignments (
    assignment_id INTEGER PRIMARY KEY AUTOINCREMENT,
    task_id INTEGER NOT NULL,
    assignee_id INTEGER NOT NULL,
    status_id INTEGER NOT NULL,
    manager_comment TEXT,
    employee_comment TEXT,
    assigned_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
    updated_at TEXT,
    FOREIGN KEY (task_id) REFERENCES tasks(task_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (assignee_id) REFERENCES users(user_id)
```

```
        ON UPDATE CASCADE
        ON DELETE RESTRICT,
    FOREIGN KEY (status_id) REFERENCES task_statuses(status_id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE IF NOT EXISTS reports (
    report_id INTEGER PRIMARY KEY AUTOINCREMENT,
    generated_by INTEGER NOT NULL,
    report_title TEXT NOT NULL,
    period_start TEXT NOT NULL,
    period_end TEXT NOT NULL,
    total_tasks INTEGER NOT NULL DEFAULT 0,
    completed_tasks INTEGER NOT NULL DEFAULT 0,
    overdue_tasks INTEGER NOT NULL DEFAULT 0,
    completion_rate REAL NOT NULL DEFAULT 0,
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (generated_by) REFERENCES users(user_id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE IF NOT EXISTS indicators (
    indicator_id INTEGER PRIMARY KEY AUTOINCREMENT,
    indicator_name TEXT NOT NULL UNIQUE,
    unit TEXT NOT NULL,
    description TEXT
);

CREATE TABLE IF NOT EXISTS report_indicators (
    report_indicator_id INTEGER PRIMARY KEY AUTOINCREMENT,
    report_id INTEGER NOT NULL,
    indicator_id INTEGER NOT NULL,
    indicator_value REAL NOT NULL,
    FOREIGN KEY (report_id) REFERENCES reports(report_id)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    FOREIGN KEY (indicator_id) REFERENCES indicators(indicator_id)
        ON UPDATE CASCADE
        ON DELETE RESTRICT
);

CREATE TABLE IF NOT EXISTS activity_logs (
    log_id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER,
    action_type TEXT NOT NULL,
    action_description TEXT NOT NULL,
    created_at TEXT NOT NULL DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
        ON UPDATE CASCADE
        ON DELETE SET NULL
);

CREATE INDEX IF NOT EXISTS idx_users_role_id
ON users(role_id);

CREATE INDEX IF NOT EXISTS idx_tasks_created_by
ON tasks(created_by);

CREATE INDEX IF NOT EXISTS idx_tasks_due_at
ON tasks(due_at);
```

```
CREATE INDEX IF NOT EXISTS idx_assignments_task_id
ON task_assignments(task_id);

CREATE INDEX IF NOT EXISTS idx_assignments_assignee_id
ON task_assignments(assignee_id);

CREATE INDEX IF NOT EXISTS idx_assignments_status_id
ON task_assignments(status_id);

INSERT OR IGNORE INTO roles(role_id, role_name, description) VALUES
(1, 'Власник бізнесу', 'Перегляд показників, контроль виконання та аналіз
звітів'),
(2, 'Менеджер', 'Створення завдань, призначення виконавців і формування
звітів'),
(3, 'Співробітник', 'Перегляд власних завдань і оновлення статусу виконання'),
(4, 'Адміністратор', 'Керування користувачами, ролями та довідниками системи');

INSERT OR IGNORE INTO task_statuses(status_id, status_code, status_name,
description) VALUES
(1, 'NEW', 'Нове', 'Завдання створене та очікує виконання'),
(2, 'IN_PROGRESS', 'У процесі', 'Завдання перебуває у виконанні'),
(3, 'DONE', 'Виконано', 'Завдання завершене виконавцем'),
(4, 'CANCELLED', 'Скасовано', 'Завдання скасоване менеджером'),
(5, 'OVERDUE', 'Прострочено', 'Завдання не виконане у встановлений строк');

INSERT OR IGNORE INTO indicators(indicator_id, indicator_name, unit,
description) VALUES
(1, 'Кількість завдань', 'шт.', 'Загальна кількість завдань за вибраний
період'),
(2, 'Кількість виконаних завдань', 'шт.', 'Кількість завдань зі статусом
виконання'),
(3, 'Кількість прострочених завдань', 'шт.', 'Кількість завдань із порушеним
строком'),
(4, 'Рівень виконання', '%', 'Відсоток виконаних завдань від загальної
кількості');
```

# ДОДАТОК Б

Програмний код серверної частини системи

```
const express = require("express");
const session = require("express-session");
const bcrypt = require("bcryptjs");
const sqlite3 = require("sqlite3").verbose();
const fs = require("fs");
const path = require("path");

const app = express();
const db = new sqlite3.Database("remote_business.db");

app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, "public")));

app.use(session({
    secret: "remote-business-secret",
    resave: false,
    saveUninitialized: false
}));

function initDatabase() {
    const schema = fs.readFileSync(path.join(__dirname, "schema.sql"), "utf8");
    db.exec(schema);

    const passwordHash = bcrypt.hashSync("admin123", 10);

    db.run(
        `INSERT OR IGNORE INTO users(user_id, role_id, full_name, email,
password_hash, phone)
         VALUES (1, 4, 'Адміністратор системи', 'admin@local.com', ?,
'+380000000000')`,
        [passwordHash]
    );
}

function run(sql, params = []) {
    return new Promise((resolve, reject) => {
        db.run(sql, params, function(error) {
            if (error) reject(error);
            else resolve(this);
        });
    });
}

function get(sql, params = []) {
    return new Promise((resolve, reject) => {
        db.get(sql, params, function(error, row) {
            if (error) reject(error);
            else resolve(row);
        });
    });
}

function all(sql, params = []) {
    return new Promise((resolve, reject) => {
        db.all(sql, params, function(error, rows) {
```

```
            if (error) reject(error);
            else resolve(rows);
        });
    });
}

function requireAuth(req, res, next) {
    if (!req.session.user) {
        return res.status(401).json({ message: "Користувач не авторизований" });
    }
    next();
}

function requireRole(roles) {
    return function(req, res, next) {
        if (!req.session.user || !roles.includes(req.session.user.role_name)) {
            return res.status(403).json({ message: "Недостатньо прав
доступу" });
        }
        next();
    };
}

async function logAction(userId, type, description) {
    await run(
        `INSERT INTO activity_logs(user_id, action_type, action_description)
         VALUES (?, ?, ?)`,
        [userId, type, description]
    );
}

app.post("/api/login", async (req, res) => {
    const { email, password } = req.body;

    const user = await get(
        `SELECT users.*, roles.role_name
         FROM users
         JOIN roles ON roles.role_id = users.role_id
         WHERE users.email = ? AND users.is_active = 1`,
        [email]
    );

    if (!user || !bcrypt.compareSync(password, user.password_hash)) {
        return res.status(400).json({ message: "Неправильний логін або
пароль" });
    }

    req.session.user = {
        user_id: user.user_id,
        full_name: user.full_name,
        email: user.email,
        role_name: user.role_name
    };

    await logAction(user.user_id, "LOGIN", "Користувач виконав вхід до
системи");

    res.json({
        message: "Авторизацію виконано успішно",
        user: req.session.user
    });
});
```

```
app.post("/api/logout", requireAuth, async (req, res) => {
    const userId = req.session.user.user_id;
    await logAction(userId, "LOGOUT", "Користувач завершив роботу із системою");

    req.session.destroy(() => {
        res.json({ message: "Вихід виконано успішно" });
    });
});

app.get("/api/dashboard", requireAuth, async (req, res) => {
    const totalTasks = await get(`SELECT COUNT(*) AS value FROM tasks`);
    const completedTasks = await get(
        `SELECT COUNT(*) AS value
         FROM task_assignments
         WHERE status_id = 3`
    );
    const overdueTasks = await get(
        `SELECT COUNT(*) AS value
         FROM task_assignments
         WHERE status_id = 5`
    );

    const total = totalTasks.value || 0;
    const completed = completedTasks.value || 0;
    const rate = total > 0 ? Number(((completed / total) * 100).toFixed(2)) : 0;

    res.json({
        total_tasks: total,
        completed_tasks: completed,
        overdue_tasks: overdueTasks.value || 0,
        completion_rate: rate
    });
});

app.get("/api/users", requireAuth, requireRole(["Адміністратор", "Власник
бізнесу", "Менеджер"]), async (req, res) => {
    const users = await all(
        `SELECT users.user_id, users.full_name, users.email, users.phone,
users.is_active, roles.role_name
         FROM users
         JOIN roles ON roles.role_id = users.role_id
         ORDER BY users.full_name`
    );

    res.json(users);
});

app.post("/api/users", requireAuth, requireRole(["Адміністратор"]), async (req,
res) => {
    const { role_id, full_name, email, password, phone } = req.body;
    const passwordHash = bcrypt.hashSync(password, 10);

    const result = await run(
        `INSERT INTO users(role_id, full_name, email, password_hash, phone)
         VALUES (?, ?, ?, ?, ?)`,
        [role_id, full_name, email, passwordHash, phone]
    );

    await logAction(req.session.user.user_id, "CREATE_USER", `Створено
користувача ${full_name}`);

    res.json({
        message: "Користувача створено",
```

```
        user_id: result.lastID
    });
});

app.get("/api/tasks", requireAuth, async (req, res) => {
    let query = `
        SELECT
            tasks.task_id,
            tasks.title,
            tasks.description,
            tasks.priority,
            tasks.due_at,
            creator.full_name AS created_by_name,
            assignee.full_name AS assignee_name,
            task_statuses.status_name
        FROM tasks
        JOIN users AS creator ON creator.user_id = tasks.created_by
        LEFT JOIN task_assignments ON task_assignments.task_id = tasks.task_id
        LEFT JOIN users AS assignee ON assignee.user_id =
task_assignments.assignee_id
        LEFT JOIN task_statuses ON task_statuses.status_id =
task_assignments.status_id
    `;

    const params = [];

    if (req.session.user.role_name === "Співробітник") {
        query += ` WHERE task_assignments.assignee_id = ?`;
        params.push(req.session.user.user_id);
    }

    query += ` ORDER BY tasks.due_at ASC`;

    const tasks = await all(query, params);
    res.json(tasks);
});

app.post("/api/tasks", requireAuth, requireRole(["Менеджер", "Власник
бізнесу"]), async (req, res) => {
    const { title, description, priority, due_at, assignee_id } = req.body;

    const taskResult = await run(
        `INSERT INTO tasks(created_by, title, description, priority, due_at)
         VALUES (?, ?, ?, ?, ?)`,
        [req.session.user.user_id, title, description, priority, due_at]
    );

    await run(
        `INSERT INTO task_assignments(task_id, assignee_id, status_id)
         VALUES (?, ?, 1)`,
        [taskResult.lastID, assignee_id]
    );

    await logAction(req.session.user.user_id, "CREATE_TASK", `Створено завдання
${title}`);

    res.json({
        message: "Завдання створено та призначено виконавцю",
        task_id: taskResult.lastID
    });
});

app.put("/api/tasks/:id/status", requireAuth, async (req, res) => {
```

```
    const { status_id, employee_comment } = req.body;
    const taskId = req.params.id;

    if (req.session.user.role_name === "Співробітник") {
        await run(
            `UPDATE task_assignments
             SET status_id = ?, employee_comment = ?, updated_at =
CURRENT_TIMESTAMP
             WHERE task_id = ? AND assignee_id = ?`,
            [status_id, employee_comment, taskId, req.session.user.user_id]
        );
    } else {
        await run(
            `UPDATE task_assignments
             SET status_id = ?, updated_at = CURRENT_TIMESTAMP
             WHERE task_id = ?`,
            [status_id, taskId]
        );
    }

    await logAction(req.session.user.user_id, "UPDATE_STATUS", `Оновлено статус
завдання №${taskId}`);

    res.json({ message: "Статус завдання оновлено" });
});

app.post("/api/reports", requireAuth, requireRole(["Менеджер", "Власник
бізнесу"]), async (req, res) => {
    const { period_start, period_end } = req.body;

    const total = await get(
        `SELECT COUNT(*) AS value
         FROM tasks
         WHERE DATE(created_at) BETWEEN DATE(?) AND DATE(?)`,
        [period_start, period_end]
    );

    const completed = await get(
        `SELECT COUNT(*) AS value
         FROM tasks
         JOIN task_assignments ON task_assignments.task_id = tasks.task_id
         WHERE task_assignments.status_id = 3
           AND DATE(tasks.created_at) BETWEEN DATE(?) AND DATE(?)`,
        [period_start, period_end]
    );

    const overdue = await get(
        `SELECT COUNT(*) AS value
         FROM tasks
         JOIN task_assignments ON task_assignments.task_id = tasks.task_id
         WHERE task_assignments.status_id = 5
           AND DATE(tasks.created_at) BETWEEN DATE(?) AND DATE(?)`,
        [period_start, period_end]
    );

    const totalValue = total.value || 0;
    const completedValue = completed.value || 0;
    const overdueValue = overdue.value || 0;
    const completionRate = totalValue > 0 ? Number(((completedValue /
totalValue) * 100).toFixed(2)) : 0;

    const report = await run(
        `INSERT INTO reports(
```

```
            generated_by, report_title, period_start, period_end,
            total_tasks, completed_tasks, overdue_tasks, completion_rate
        )
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)`,
        [
            req.session.user.user_id,
            "Звіт про виконання завдань",
            period_start,
            period_end,
            totalValue,
            completedValue,
            overdueValue,
            completionRate
        ]
    );

    await logAction(req.session.user.user_id, "CREATE_REPORT", "Сформовано звіт
про виконання завдань");

    res.json({
        message: "Звіт сформовано",
        report_id: report.lastID,
        total_tasks: totalValue,
        completed_tasks: completedValue,
        overdue_tasks: overdueValue,
        completion_rate: completionRate
    });
});

initDatabase();

app.listen(3000, () => {
    console.log("Сервер запущено: http://localhost:3000");
});
```

# ДОДАТОК В

Програмний код клієнтської частини та запуску системи

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>ІС дистанційного керування бізнесом</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <main class="layout">
        <section class="card" id="login-panel">
            <h1>Вхід до системи</h1>
            <input id="email" type="email" placeholder="Email">
            <input id="password" type="password" placeholder="Пароль">
            <button onclick="login()">Увійти</button>
            <p id="login-message"></p>
        </section>

        <section class="dashboard hidden" id="app-panel">
            <header class="topbar">
                <div>
                    <h1>Дистанційне керування бізнесом</h1>
                    <p id="user-info"></p>
                </div>
                <button onclick="logout()">Вийти</button>
            </header>

            <section class="stats">
                <div class="stat">
                    <span>Усього завдань</span>
                    <strong id="total-tasks">0</strong>
                </div>
                <div class="stat">
                    <span>Виконано</span>
                    <strong id="completed-tasks">0</strong>
                </div>
                <div class="stat">
                    <span>Прострочено</span>
                    <strong id="overdue-tasks">0</strong>
                </div>
                <div class="stat">
                    <span>Рівень виконання</span>
                    <strong id="completion-rate">0%</strong>
                </div>
            </section>

            <section class="grid">
                <div class="card">
                    <h2>Створення завдання</h2>
                    <input id="task-title" placeholder="Назва завдання">
                    <textarea id="task-description" placeholder="Опис
завдання"></textarea>
                    <select id="task-priority">
                        <option>Низький</option>
                        <option selected>Середній</option>
                        <option>Високий</option>
```

```
                    </select>
                    <input id="task-due" type="date">
                    <select id="task-assignee"></select>
                    <button onclick="createTask()">Створити завдання</button>
                </div>

                <div class="card">
                    <h2>Формування звіту</h2>
                    <input id="period-start" type="date">
                    <input id="period-end" type="date">
                    <button onclick="createReport()">Сформувати звіт</button>
                    <pre id="report-result"></pre>
                </div>
            </section>

            <section class="card">
                <h2>Список завдань</h2>
                <table>
                    <thead>
                        <tr>
                            <th>Назва</th>
                            <th>Пріоритет</th>
                            <th>Строк</th>
                            <th>Виконавець</th>
                            <th>Статус</th>
                            <th>Дія</th>
                        </tr>
                    </thead>
                    <tbody id="tasks-body"></tbody>
                </table>
            </section>
        </section>
    </main>

    <script src="app.js"></script>
</body>
</html>
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background: #f4f6f8;
    color: #1f2933;
}

.layout {
    width: 1100px;
    max-width: 95%;
    margin: 30px auto;
}

.hidden {
    display: none;
}

.card {
    background: #ffffff;
    border: 1px solid #d9e2ec;
    border-radius: 10px;
    padding: 20px;
    margin-bottom: 18px;
}

.topbar {
```

```
    display: flex;
    justify-content: space-between;
    align-items: center;
    background: #ffffff;
    border: 1px solid #d9e2ec;
    border-radius: 10px;
    padding: 20px;
    margin-bottom: 18px;
}

.stats {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 16px;
    margin-bottom: 18px;
}

.stat {
    background: #ffffff;
    border: 1px solid #d9e2ec;
    border-radius: 10px;
    padding: 18px;
}

.stat span {
    display: block;
    color: #627d98;
    margin-bottom: 8px;
}

.stat strong {
    font-size: 26px;
}

.grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 18px;
}

input, select, textarea, button {
    width: 100%;
    box-sizing: border-box;
    margin: 7px 0;
    padding: 10px;
    border: 1px solid #bcccdc;
    border-radius: 6px;
    font-size: 14px;
}

button {
    background: #1f2933;
    color: #ffffff;
    cursor: pointer;
}

button:hover {
    background: #323f4b;
}

table {
    width: 100%;
    border-collapse: collapse;
```

```
}

th, td {
    border-bottom: 1px solid #d9e2ec;
    padding: 10px;
    text-align: left;
}

th {
    background: #f0f4f8;
}
let currentUser = null;

async function request(url, options = {}) {
    const response = await fetch(url, {
        headers: { "Content-Type": "application/json" },
        ...options
    });

    const data = await response.json();

    if (!response.ok) {
        throw new Error(data.message || "Помилка виконання запиту");
    }

    return data;
}

async function login() {
    const email = document.getElementById("email").value;
    const password = document.getElementById("password").value;

    try {
        const data = await request("/api/login", {
            method: "POST",
            body: JSON.stringify({ email, password })
        });

        currentUser = data.user;

        document.getElementById("login-panel").classList.add("hidden");
        document.getElementById("app-panel").classList.remove("hidden");
        document.getElementById("user-info").textContent =
            `${currentUser.full_name}, роль: ${currentUser.role_name}`;

        await loadUsers();
        await loadDashboard();
        await loadTasks();
    } catch (error) {
        document.getElementById("login-message").textContent = error.message;
    }
}

async function logout() {
    await request("/api/logout", { method: "POST" });
    location.reload();
}

async function loadDashboard() {
    const data = await request("/api/dashboard");

    document.getElementById("total-tasks").textContent = data.total_tasks;
    document.getElementById("completed-tasks").textContent =
```

```
data.completed_tasks;
    document.getElementById("overdue-tasks").textContent = data.overdue_tasks;
    document.getElementById("completion-rate").textContent = `$
{data.completion_rate}%`;
}

async function loadUsers() {
    try {
        const users = await request("/api/users");
        const select = document.getElementById("task-assignee");

        select.innerHTML = "";

        users
            .filter(user => user.role_name === "Співробітник")
            .forEach(user => {
                const option = document.createElement("option");
                option.value = user.user_id;
                option.textContent = user.full_name;
                select.appendChild(option);
            });
    } catch (error) {
        console.log("Список користувачів недоступний для цієї ролі");
    }
}

async function loadTasks() {
    const tasks = await request("/api/tasks");
    const body = document.getElementById("tasks-body");

    body.innerHTML = "";

    tasks.forEach(task => {
        const row = document.createElement("tr");

        row.innerHTML = `
            <td>${task.title}</td>
            <td>${task.priority}</td>
            <td>${task.due_at}</td>
            <td>${task.assignee_name || "-"}</td>
            <td>${task.status_name || "Нове"}</td>
            <td>
                <select onchange="updateStatus(${task.task_id}, this.value)">
                    <option value="">Змінити статус</option>
                    <option value="1">Нове</option>
                    <option value="2">У процесі</option>
                    <option value="3">Виконано</option>
                    <option value="4">Скасовано</option>
                    <option value="5">Прострочено</option>
                </select>
            </td>
        `;

        body.appendChild(row);
    });
}

async function createTask() {
    const payload = {
        title: document.getElementById("task-title").value,
        description: document.getElementById("task-description").value,
        priority: document.getElementById("task-priority").value,
        due_at: document.getElementById("task-due").value,
```

```
        assignee_id: document.getElementById("task-assignee").value
    };

    await request("/api/tasks", {
        method: "POST",
        body: JSON.stringify(payload)
    });

    document.getElementById("task-title").value = "";
    document.getElementById("task-description").value = "";

    await loadDashboard();
    await loadTasks();
}

async function updateStatus(taskId, statusId) {
    if (!statusId) return;

    await request(`/api/tasks/${taskId}/status`, {
        method: "PUT",
        body: JSON.stringify({
            status_id: Number(statusId),
            employee_comment: "Статус оновлено через інтерфейс системи"
        })
    });

    await loadDashboard();
    await loadTasks();
}

async function createReport() {
    const period_start = document.getElementById("period-start").value;
    const period_end = document.getElementById("period-end").value;

    const report = await request("/api/reports", {
        method: "POST",
        body: JSON.stringify({ period_start, period_end })
    });

    document.getElementById("report-result").textContent =
        `Звіт №${report.report_id}
Усього завдань: ${report.total_tasks}
Виконано: ${report.completed_tasks}
Прострочено: ${report.overdue_tasks}
Рівень виконання: ${report.completion_rate}%`;

    await loadDashboard();
}
```

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

**Декларація про академічну доброчесність та використання ШІ**

## ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Білоголовий Назар Сергійович, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Інформаційна система дистанційного керування бізнесом» є результатом моєї особистої праці.
2. Усі запозичення з текстів інших авторів мають відповідні посилання згідно з чинними стандартами.
3. Щодо використання інструментів ШІ зазначаю, що (обрати необхідне):
    - ○ [ ] інструменти генеративного ШІ не використовувалися.
    - ○ [+] інструменти ШІ (вказати назву: ChatGPT, Gemini, Claude, DeepL, _____інші (вказати назву)) були використані для:
        - ■ [ ] перекладу іншомовних джерел;
        - ■ [ ] стилістичного редагування та перевірки граматики;
        - ■ [+] допомоги у написанні/відлагодженні програмного коду;
        - ■ [ ] інше: _____.

Я розумію, що надання недостовірної інформації може призвести до скасування результатів захисту та відрахування з числа здобувачів НУБіП України.

«24» травня 2026 р. _____ **Білоголовий Назар**
(підпис)